Ein Dekompositionsansatz für das taktisch-operative Management von Bike-Sharing-Systemen

Von der Carl-Friedrich-Gauß-Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades einer
Doktorin der Wirtschaftswissenschaften (Dr. rer. pol.)

genehmigte Dissertation

von
Viola Hsiao-Han Ricker
geboren am 12.05.1985
in Arolsen, jetzt Bad Arolsen

Eingereicht am:	19.11.2015
Disputation am:	14.12.2015
1. Referent:	Prof. Dr. Dirk C. Mattfeld
2. Referent:	Prof. Dr. Jan F. Ehmke

(2015)

Zusammenfassung

Die Mobilitätsbedürfnisse der Menschen in Großstädten steigen und mit ihnen auch Straßenauslastung und Umweltverschmutzung. Bike-Sharing-Systeme bieten eine nachhaltige Alternative für den öffentlichen Personennahverkehr, die einige der Verkehrsprobleme entschärfen könnte. Dafür bedarf es großer Nutzerzahlen, die durch entsprechende Servicequalität erreicht werden können. Dabei besteht die Herausforderung darin, dass dem im Betrieb entstehenden Ungleichgewicht in der räumlichen Verteilung der Fahrräder im System entgegengewirkt wird. Damit den Kunden zu jeder Zeit Fahrräder und Fahrradstellplätze zur Verfügung stehen, müssen Maßnahmen zur Repositionierung der Fahrräder getroffen werden. Diese Maßnahmen bestehen aus dem Transport der Räder von Stationen mit Stellplatzbedarf zu Stationen mit Fahrradbedarf. Die vorliegende Arbeit untersucht das Problem der Repositionierung und stellt ein Konzept eines dekomponierten Optimierungsverfahrens vor, welches mittels antizipierender stochastischer und dynamischer Optimierung die Menge der für die Repositionierung relevanten Stationen ermittelt. Auf Basis dieses Ergebnisses schließt sich die heuristische Tourenplanung an.

Danksagung

Auf dem Weg von der Forschungsidee bis hin zur fertigen Dissertation begleiteten mich viele Menschen, ohne die das ganze Projekt nicht möglich gewesen wäre.
Allen voran danke ich meinem Betreuer Professor Dirk Mattfeld für die Ermöglichung meiner Forschung. Die lückenlose Unterstützung durch fachliche Diskussionen und Anregungen war bei der Umsetzung vieler Ideen enorm hilfreich.
Vielen Dank an meinen zweiten Gutachter Professor Jan Ehmke für die Begleitung während meiner Zeit am Lehrstuhl, vom ersten Tag an bis heute. Danke für das Teilen der Expertise in vielen Gesprächen und das Interesse an meiner Forschung.
Mein Lehrstuhlteam war zu jeder Zeit und in jeder Besetzung eine große Stütze. Danke Euch allen für die offenen Türen und Ohren, für hilfreiche Anmerkungen, Korrekturen oder das Teilen von Erfahrung. Und auch über die Lehrstuhlgrenzen hinaus geht mein Dank an Kollegen und Freunde, die mir auf ihren speziellen Fachgebieten oder auch als mentale Unterstützer eine riesige Hilfe waren.
Schließlich geht ein großer Teil meiner Dankbarkeit an Familie und Freunde außerhalb des universitären Lebens. Die wichtigsten Personen sind hier meine Mama und meine Schwestern, die immer an mich geglaubt und zur richtigen Zeit die richtigen Motivationspäckchen geschickt haben. Und natürlich mein Freund Jan, der als Fels in der Brandung viele Berg- und Talfahrten meiner Forschung direkt miterlebt und überstanden hat. Und last but not least bleibt den vielen guten Freunde zu danken, die immer da waren, sich mit über gute Zeiten gefreut haben und schlechte Zeiten durch liebe Worte, eine Umarmung oder Schokolade besser zu machen wussten. Jeder von Euch hat dabei geholfen, dass ich diese Arbeit zu dem machen konnte, was sie jetzt ist.

Viola Ricker

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Algorithmenverzeichnis

1. Einleitung

1.1. Motivation

Mit den steigenden Mobilitätsbedürfnissen und der Tendenz hin zu immer größeren Städten rückt in den letzten Jahren immer mehr die Frage nach nachhaltigen Alternativen im Verkehr in den Fokus der Öffentlichkeit. Europa- und weltweite Studien [30] [78] zeigen Verkehrsprobleme wie Emissionen oder Staus auf und prognostizieren deren Eskalation in der Zukunft. Als mögliche Auswege werden der Ausbau und die Weiterentwicklung des öffentlichen Personennahverkehrs (ÖPNV) vorgeschlagen. Als eine „grüne“ Alternative werden dabei die immer populärer werdenden Bike-Sharing-Systeme (BSS) sowie ein Ausbau der Fahrradinfrastruktur hervorgehoben. Zwischen den Jahren 2000 und 2010 ist die Anzahl der Städte mit BSS von einstellig auf fast 400 gestiegen. Mittlerweile haben sie also nicht mehr den Charakter eines interessanten Experiments, sondern haben sich im ÖPNV als echte Alternative manifestiert [51]. Die gesamte Flotte an öffentlichen Rädern hat inzwischen eine Größe von über 400000 erreicht [32]. Die häufigste Form des BSS ist die stationsbasierte, wie sie in vielen Großstädten zum Einsatz kommt. Prominente Beispiele sind die Systeme vélib in Paris [1] oder citibike in New York [54]. In diesen werden die Fahrräder dem Kunden an Stationen zur Verfügung gestellt, die in der Stadt verteilt sind. Die Stationen haben eine bestimmte Kapazität an Fahrradstellplätzen, auch Boxen genannt, an denen die Räder automatisiert entnommen sowie zurückgegeben werden können. Es ist weder eine Reservierung noch eine Angabe der Zielstation notwendig. Die Kunden genießen dadurch eine hohe Flexibilität.
Damit die Vorteile bezüglich der Nachhaltigkeit von Sharing-Systemen zum Tragen kommen, sind große Nutzerzahlen ausschlaggebend [53]. Folglich ist es ein wichtiges Ziel der BSS-Betreiber, Akzeptanz in der Bevölkerung zu schaffen und so das System zu vergrößern. Dabei hängt die Akzeptanz stark von der Zufriedenheit der Nutzer ab. Diese kann der Betreiber durch reibungslosen Service, also das Ermöglichen von problemlosen Entleihungen und Rückgaben von Rädern, erreichen.

Durch Spontanität und Flexibilität auf der Kundenseite unterliegt die Nachfrage nach Fahrrädern und Stellplätzen einer großen Dynamik und Stochastik. Beides erschwert die Sicherstellung eines hinreichend guten Services. Im System CityBike in Wien [35] dauern beispielsweise 90% der Fahrten weniger als eine Stunde, wobei die Medianfahrtdauer bei 15 Minuten liegt [80]. So kann in kurzer Zeit ein Ungleichgewicht in der Verteilung der Räder über die Stationen entstehen. Falls die Fahrräder in den Stationen ausgehen oder sämtliche Fahrradstellplätze belegt sind, können neue Anfragen von Kunden nach Fahrrädern oder Stellplätzen an den entsprechenden Stationen nicht erfüllt werden. Dies führt dazu, dass Kunden Umwege auf sich nehmen müssen, oder sich von der Nutzung des Systems abwenden. Im Sinne des Systemwachstums sind solche Situationen also unbedingt zu vermeiden.

Maßnahmen gegen ungünstige Verteilungen der Räder im System können in drei Ebenen eingeordnet werden. Die *strategische* Ebene beschäftigt sich mit dem Systemdesign. Durch das Hinzufügen neuer oder den Ausbau vorhandener Stationen können hier Kapazitätsprobleme vermindert werden. Auf *taktischer* Ebene wird über die Anzahl und Verteilung der Räder auf die Stationen entschieden. An beliebten Ausgangsstationen sollten mehr Fahrräder zur Verfügung stehen, während an beliebten Zielorten mehr Stellplätze frei sein sollten. Die *operative* Ebene ist für die manuelle Umverteilung der Räder von Betreiberseite zuständig. Mittels sogenannter Repositionierungsfahrten können Fahrradtransporter Räder von Stationen mit Stellplatzbedarf zu Stationen mit Fahrradbedarf bringen. So kann kurzfristig auf kundenseitige Bewegungen reagiert werden.

Der Fokus der vorliegenden Arbeit liegt sowohl auf der taktischen, als auch auf der operativen Ebene. Simultan werden sowohl die Fahrradverteilung über das System als auch die entsprechend notwendigen Repositionierungen geplant. Die Komplexität dieser Problemstellung wird sofort ersichtlich, wenn die Fragestellung verbalisiert wird: **Wie viele** Fahrräder sollen **wann** und von welchen **Pickup-Stationen** auf **welchem Transportfahrzeug** zu welchen **Delivery-Stationen** transportiert werden, um dort einen günstigen **Bestand** zu erzeugen? Es handelt sich hier also um ein Inventory Routing Problem (IRP) mit Pickups und Deliveries. Die Dynamik und die stochastische Kundennachfrage erschweren das Problem, welches schon in einfachster Form ohne Betrachtung der Bestände an den Stationen (Vehicle Routing Problem mit einem Fahrzeug) in die Komplexitätsklasse $\mathcal{NP}$ fällt [44], zusätzlich.

Das vermehrte Interesse an BSS ist auch in der wissenschaftlichen Literatur er-

kennbar. In den letzten Jahren sind aus vereinzelten Artikeln eine große Sammlung von Forschungsbeiträgen in verschiedenen Themengebieten rund um BSS geworden. Die historische Entwicklung der BSS sowie deren Einflüsse auf den Verkehr, die Menschen und die Umwelt werden von DeMajo [23] und Shaheen et al [72] zusammengefasst. Pucher und Buehler tragen in ihrem Buch [60] viele Beiträge zum Radfahren in Städten zusammen. Dabei sind die Themen der einzelnen Kapitel weit gestreut und befassen sich mit den Trends rund um den Globus, Vorteilen bzgl. Verkehr und Gesundheit sowie Sicherheit und der zugehörigen Radfahrausstattung. Ein ausführlicherer Überblick über die Forschung zu BSS, insbesondere zur Optimierung von BSS wird in Kapitel 2.1 gegeben.

1.2. Gliederung der vorliegenden Arbeit

Die Arbeit ist in den folgenden Kapiteln wie folgt gegliedert:
Kapitel 2 führt in die Thematik des Bike-Sharings ein und enthält ebenso die Grundlagen zu Dekompositionen sowie Approximativer Dynamischer Programmierung. In Kapitel 3 erfolgt die mathematische Problemformulierung. Kapitel 4 dient der Einordnung in den Kontext betrieblicher Informationssysteme. Mittels eines Ebenenmodells in Analogie zu Informationsflussmodellen im Unternehmenskontext werden die verschiedenen Ebenen der Planungsvorgänge dargestellt. Die Problemkomplexität motiviert die Entwicklung eines angepassten Dekompositionsverfahren, welches im Anschluss in Kapitel 5 ausführlich beschrieben wird. Die Erläuterungen zum experimentellen Aufbau, basierend auf einer Bike-Sharing-Simulation, erfolgt in Kapitel 6. Kapitel 7 fasst die Erkenntnisse der Arbeit zusammen und gibt einen Ausblick auf mögliche Weiterführungen der Forschung.

2. Grundlagen und Definitionen

Dieses Kapitel enthält zunächst die Einführung in die Thematik des Bike-Sharings und der Herausforderungen, denen sich ein BSS-Betreiber stellt. Kapitel 2.2 und 2.3 legen dann die methodischen Grundlagen für die Dekomposition und die Approximative Dynamische Programmierung.

2.1. Bike-Sharing

2.1.1. Geschichte

Das Bike-Sharing lässt sich in seiner inzwischen 50 Jahre umfassenden Geschichte hinsichtlich des Aufbaus seiner Systeme in drei Generationen unterteilen [23]. Die erste Generation ist durch Versuche wie die White Bikes in Amsterdam im Jahr 1965 geprägt. Dort wurden namensgebend weiß lackierte Räder der Öffentlichkeit kostenlos zur Verfügung gestellt. Diebstahl und Vandalismus brachten das System allerdings schon nach wenigen Tagen zum Einsturz. Die ersten Systeme der zweiten Generation gab es erst Anfang der 90er Jahre in Dänemark. Dort kamen Fahrradstationen zum Einsatz, an denen die Räder gegen ein Münzpfand entliehen werden konnten. Trotz einiger Verbesserungen bezüglich der Robustheit der Räder konnten sich diese Systeme nicht vor Diebstahl schützen, da die Nutzer auch hier anonym blieben. Die dritte Generation des Bike-Sharings entwickelte sich zum Ende der 90er Jahre. Die Stationen wurden moderner und durch Informationssysteme unterstützt. Eine Anmeldung per Nutzerkarte nahm den Nutzern die Anonymität, automatisierte Entleihungen und Rückgaben sowie die teilweise angebotenen Freiminuten [35] machten die Nutzung attraktiver. Es wird vorhergesagt, dass eine vierte Generation des Bike-Sharing den Markt erobern wird. Auszeichnen soll sie eine verbesserte Effizienz, Nachhaltigkeit und Komfort. Zum Beispiel wäre die Erweiterung der Flotte durch E-Fahrzeuge und die zugehörige Infrastruktur denkbar. Es könnten außerdem neue Geschäftsmodelle zum Einsatz kommen. In diesem Zusammenhang ist auch die Optimierung des Betriebsablaufs wichtig.

In den letzten Jahren wuchs die Anzahl der Systeme gewaltig. Ende 2014 betrieben 855 Städte auf der Welt Bike-Sharing-Systeme [56]. China hält mit 73 neuen Bike-Sharing-Städten den Rekord an Systemanzahl und auch Fahrradflottengröße.

2.1.2. Vorteile

Die Vorteile von BSS sehen Shaheen et al. [72] in folgenden Punkten:

- umweltschonend
- kostengünstig
- flexibel
- gesund
- steigert die Beliebtheit des Fahrrads als Transportmittel

Es gibt einige Quellen, welche Umfrageergebnisse anführen, die Emissionsreduktion belegen. Nach Fishman et al. [32] sind für solche Aussagen allerdings noch weitere Untersuchungen notwendig. Angenommen werden kann jedoch, dass Vorteile hinsichtlich der Umwelt nur zum Tragen kommen, wenn die Nutzerzahlen entsprechend groß sind [53].

2.1.3. Herausforderungen

Der reibungslose Betriebsablauf von BSS hängt von vielen Faktoren ab [51]. Um die Nutzung attraktiv zu machen, ist es wichtig, sie einfach zu gestalten. Dazu gehört der Ausbau des Fahrradwegenetzes, um das BSS gut erreichbar in das vorhandene Verkehrsnetz zu integrieren. Zudem ist eine schnelle und unkomplizierte Anwendung vorteilhaft. Dem gegenüber steht die Sicherheit. Missbrauch und Diebstahl lassen sich nur verhindern, wenn die Nutzer sich identifizieren müssen, bevor sie eine Entleihung vornehmen können. Moderne Informationssysteme können diesen Vorgang beschleunigen und vereinfachen. Materialrobustheit sowie -sicherheit spielen eine weitere große Rolle. Sie stärken das Vertrauen der Nutzer in das System und reduzieren die Wartungsarbeiten. Ein großer Vorteil, den BSS ihren Nutzern im Gegensatz zu traditionellen Fahrradvermietungen bieten, ist Flexibilität. Ohne Reservierung ist das Entleihen an allen Stationen spontan möglich und Einwegfahrten zu einer beliebigen anderen Station sind erlaubt. Da die Nachfrage an verschiedenen Orten im Tagesverlauf schwanken kann und die Füllstände an den Stationen

sich in der Regel nicht von selbst ausgleichen, kann es Zeiten geben, zu denen Stationen voll oder leer sind [80]. In solchen Situationen kann es zu nutzerseitigen Unannehmlichkeiten kommen. Es könnten zum Beispiel Umwege erforderlich sein oder ein spontanes Umschwenken auf ein anderes Verkehrsmittel. Dies kann dazu führen, dass die Nutzer dem System den Rücken kehren. Deshalb ist ein Systembetreiber bestrebt, Maßnahmen zu ergreifen, um seinen Service, die reibungslose Entleihung und Rückgabe der Räder, anbieten zu können. Die Planung solcher Maßnahmen ist sehr komplex und findet immer unter unvollkommenen Informationen statt. Daher ist es sinnvoll, sie nach ihren zeitlichen Reichweiten getrennt zu betrachten. Klein und Scholl [42] unterscheiden lang-, mittel- und kurzfristige Planung und ordnen diese jeweils der strategischen, taktischen oder operativen Ebene zu. Für die Planung von Frachttransporten nutzen Crainic und Laporte [20] ebenfalls die Einteilung in strategisch, taktisch und operativ. In seinem Buch „Service Network Design of Bike Sharing Systems - Analysis and Optimization“ [79] beschreibt Vogel das Service Network Design von BSS und leitet die zugehörigen logistischen Planungsebenen her. Die Maßnahmen zur Verbesserung des Services im BSS auf den einzelnen Ebenen kann man folgendermaßen beschreiben:

Strategische Maßnahmen

Das Systemdesign, das die Standorte und die Größe der Stationen beschreibt, muss auf die Bedürfnisse der Nutzer zugeschnitten sein. Das Systemnetz ist je nach Nutzung mehr oder weniger dicht zu gestalten. In der Innenstadt ist eine höhere Anzahl an Stationen möglicherweise angebrachter als in den Außenbezirken. Hoch frequentierte Stationen sollten auch dementsprechend viele Stellplätze und Fahrräder bereitstellen, damit zu Hochzeiten am Tag, zum Beispiel in der Rushhour, ein Puffer vorhanden ist.
Das Problem auf der strategischen Ebene ist ein Standortproblem. Entschieden wird über Lage und Größe der Stationen im System. Grundlagen zu Standortmodellen sind unter anderem in „Logistiknezwerke“ von Mattfeld und Vahrenkamp [48] zu finden.

Taktische Maßnahmen

Die taktische Optimierungsebene hat zur Aufgabe, die Füllstände der Stationen so festzulegen, dass sie zu jeder Tageszeit die Nachfrage der Kunden erfüllen können. Je nachdem, wie die verschiedenen Stationen charakterisiert werden [80], können sie über den Tag verteilt sehr verschiedene optimale Füllstände haben. Betrachten wir eine Station im Wohngebiet und eine andere in der Nähe eines Bürokomplexes. Falls die Stationen vorzugsweise von Berufspendlern genutzt werden, so ist morgens im Wohngebiet ein hoher Füllstand vorteilhaft, während im Büroviertel ein niedriger Füllstand besser ist. Umgekehrt verhält sich die Situation abends, wenn die Pendler wieder nach Hause fahren.
Das Problem der taktischen Ebene fällt in die Klasse der Inventory Probleme (IP). Einen Einblick in die Theorie des IP geben zum Beispiel Hax und Candea [37]. Die Entscheidung ist über die Anzahl der Räder an den Stationen zu treffen. Das Ziel ist, die Nachfrage der Kunden zu befriedigen. Der Unterschied zum klassischen IP besteht in den zwei unterschiedlichen und direkt abhängigen Nachfragen nach Rädern und Stellplätzen, die zwei Kapazitätsgrenzen mit sich bringen.

Operative Maßnahmen

Die Möglichkeit, kurzfristig in das System einzugreifen, bietet sich auf der operativen Ebene durch sogenannte Repositionierungen. Damit sind Transporte von Rädern von Stationen mit Stellplatzmangel zu Stationen mit Fahrradmangel gemeint. Die Repositionierungen müssen entsprechend in Touren kombiniert werden, um an den Stationen möglichst günstige Füllstände, wie auf der taktischen Ebene optimiert, herzustellen. Eine weniger direkt steuerbare Repositionierungsmaßnahme kann auf Kundenseite erfolgen. Durch Rabatte oder andere Anreize können Kunden dazu verleitet werden, Stationen mit Radbedarf anzufahren. Wegen der schlechten Steuerbarkeit der kundenseitigen Maßnahmen seien diese im Folgenden außer Acht gelassen.
Die Problemklasse, in die die Planung der Repositionierungstouren gehört, ist die der Vehicle Routing Probleme (VRP) [14]. Das VRP ist die Verallgemeinerung des bekannten Travelling Salesman Problems (TSP, siehe [55]) auf den Fall mit mehreren Touren. Das Ziel des TSP ist es, eine möglichst kurze Rundtour von einem Startpunkt zu allen Zielknoten und zurück zu finden. Das VRP erweitert das Problem in der Hinsicht, dass mehrere Rundtouren geplant werden, die in Summe

minimiert werden. Im BSS-Kontext ist das Ziel ein Tourenplan, der die Stationen mit Rad- oder Stellplatzbedarf abdeckt. Die zwei entgegengesetzten Bedarfe führen dazu, dass es sowohl Pickup- als auch Delivery-Stationen gibt. Entscheidungsvariablen sind die Wahl des Transporters und die Reihenfolge, in der die Stationen besucht werden. In der Literatur ist dieses Problem als one-commodity Pickup-and-Delivery VRP (1-PDVRP) bekannt [47].

Abhängigkeiten zwischen den Ebenen

Die Abhängigkeiten zwischen den drei Ebenen sind leicht nachvollziehbar. Ein großzügiges Systemdesign wird sicherlich den Aufwand auf operativer und taktischer Ebene senken. Durch größere Stationen mit einer größeren Anzahl an Rädern und Stellplätzen besteht seltener die Notwendigkeit Füllstände anzupassen und dementsprechend sind weniger oder kürzere Repositionierungstouren ausreichend. Die taktische und operative Ebene sind jedoch noch fester verknüpft. In die Entscheidung über die Füllstände sollte auch der Aufwand für die entsprechende Umsetzung einbezogen werden. Die Routenplanung auf der anderen Seite braucht den Input über die zu realisierenden Füllstände der taktischen Planung. Da die Umsetzung der optimalen Füllstände auf der operativen Ebene durch Unzulässigkeiten im Allgemeinen nicht umsetzbar ist, müssen die beiden Ebenen kommunizieren. Eine Einzelbetrachtung der beiden Ebenen ist folglich nicht anzuraten.
Kombiniert man die Problemstellungen der taktischen und operativen Ebene, befindet man sich in der Problemklasse der Inventory Routing Probleme (IRP). Ein Überblick über die Forschung im Bereich IRP geben Coelho et al. [15]. Zusätzlich zu den Entscheidungsvariablen des 1-PDVRP kommen im Falle des IRP noch die jeweiligen Entscheidungsvariablen des IP hinzu. In Kombination ergibt sich also das Ziel, einen Tourenplan zu entwerfen, dessen Pickups und Deliveries die gewünschten Fahrradbestände an den Stationen liefern, während der Aufwand für die Touren minimiert wird. Eine mathematische Formulierung des Problems im Kontext der Repositionierung erfolgt in Kapitel 3.

Bewertung der Maßnahmen

Um die verschiedenen Maßnahmen zu bewerten, müssen ihre Auswirkungen aus verschiedenen Blickwinkeln betrachtet werden. Für den Betreiber fallen durch den

Ausbau der Stationen oder den Einsatz von Repositionierungsfahrzeugen Kosten an. Diese sind numerisch darstellbar als Fahrtzeiten, Arbeitszeiten oder eine Menge von Geldeinheiten und somit leicht messbar. Aus Kundensicht wird durch die Maßnahmen der Service verbessert. Folglich sollte die Anzahl an fehlgeschlagenen Entleihungen oder Rückgaben sinken und sich die Zufriedenheit erhöhen. Diese Zufriedenheit zu messen und dann in Beziehung zu den betreiberseitigen Kosten zu stellen, ist eine große Herausforderung. Ab den 70er Jahren wurden Modelle zu Inventory Problemen extensiv diskutiert. Für Fehlbestände wurden die Variablen backorders (Lieferrückstand) für Bedarf, der nachgelagert erfüllt werden kann und lost sales (verlorene Nachfrage) für Bedarf, der nicht verzögert erfüllt werden kann, eingeführt [52]. Beide Variablen benötigen einen Strafkostensatz, der pro Einheit anfällt. Bei backorders kann das z.B. ein Rabatt sein, der für die Verspätung gewährt wird. Lost sales führen zu entgangenen Gewinnen und können in einigen Fällen auch Entschädigungszahlungen nach sich ziehen. Unzufriedenheit der Kunden oder andere Folgen, wie ein schlechter Ruf werden in diesen Modellen nicht betrachtet. Sie lassen sich nicht numerisch ausdrücken und sind schwierig vergleichbar mit den zuvor beschriebenen Kosten. Im Kontext des Bike-Sharing sind lost sales gleichzusetzen mit nicht erfolgreichen Entleihungen, bei denen die Kunden auf andere Verkehrsmittel umsteigen und so dem System den Rücken kehren. Mit backorders sind Situationen vergleichbar, bei denen Kunden kein Rad oder keinen leeren Stellplatz vorfinden und zur nächsten Station gehen oder fahren, um dort ein Rad oder einen Stellplatz vorzufinden. Beides lässt sich allerdings in diesem Umfeld nicht ohne Probleme messen. Noch schwieriger ist die numerische Formulierung, um sie mit den betreiberseitigen Kosten vergleichbar zu machen. Um die lost sales zu erfassen, könnte man mit enormen Aufwand die Kunden zählen, welche zu den Stationen kommen, um sie erfolglos wieder zu verlassen. Dies würde jedoch die Kunden vernachlässigen, die evtl. schon aus der Ferne die leere Station entdecken und sofort umkehren. Ähnlich verhält es sich mit den backorders. Es könnten natürlich Daten durch Zählen erhoben werden, allerdings würde man z.B. bei vorbeifahrenden Fahrradfahrern nicht zwischen denen entscheiden können, die erkannt haben, dass die Station voll ist, und zur nächsten müssen und denen, die sowieso ein anderes Ziel haben. Die Unzufriedenheit der Kunden könnte durch Umfragen ermittelt werden. Auch dies würde einen unverhältnismäßig hohen Aufwand mit sich bringen und zudem ein hohes Maß an Ungenauigkeit durch Subjektivität auf Kundenseite. Angenommen, die Anzahlen der lost sales und die der backorders wären bekannt, gäbe es noch die Schwierigkeit der numerischen Bewertung. Kunden,

welche durch einen geringen Umweg, vielleicht sogar in Sichtweite, eine Station zur Befriedigung ihrer Nachfrage finden, sind sicherlich weniger unzufrieden als jene, die einige Kilometer Weg auf sich nehmen müssen. Aus diesem Grund wird in dieser Arbeit ein Ansatz zur Bewertung der Servicequalität vorgeschlagen, welcher von den klassischen Ansätzen abweicht. Statt der lost sales und der backorders werden die Umwege betrachtet, die Kunden durch fehlgeschlagene Serviceleistungen auf sich nehmen müssen. Das ist in der Regel der Weg zur nächsten Station sofern dort ein Rad bzw. ein Stellplatz vorgefunden wird. Falls sie dort weiterhin nicht fündig werden, so fällt erneut ein Umweg zur nächsten Station an. Es wird davon ausgegangen, dass die Unzufriedenheit der Kunden mit der Länge der Umwege steigt. Umwege werden bei einer leeren Station zu Fuß oder bei einer vollen Station auf dem Rad zurückgelegt. Durch ihre Fahrtzeit per Rad bzw. die Zeit, die zu Fuß benötigt wird, sind sie durch Zeiteinheiten beschreibbar. Dies hat zusätzlich den Vorteil, dass die Vergleichbarkeit mit betreiberseitigen Repositionierungsfahrten direkt durch die Zeit gegeben ist. Im weiteren Verlauf der Arbeit wird folglich der Begriff *Kundenumwege* zur Evaluation der Servicequalität verwendet.

2.1.4. Forschung zu BSS in der Literatur

Mit dem rasanten Wachstum des Bike-Sharings rund um die Welt (siehe [32] und [56]) ist auch das Interesse gewachsen, die verschiedenen Herausforderungen zu untersuchen und sich ihnen zu stellen. Zu einem besseren Verständnis der Vorgänge und Bewegungen in BSS tragen unter anderem Vogel et al. [80] und Côme und Oukhellou [18] bei. Sie untersuchen die Systeme in Wien bzw. Paris und visualisieren jeweils die Nutzung. Beide Arbeiten wenden Clusterverfahren an, um die Stationen in sogenannte Nutzungsprofile einzuteilen. Diese hängen von Faktoren wie geographischer Lage und Umfeld ab. So wird ersichtlich, aus welchen unterschiedlichen Quellen die Ungleichgewichte der Fahrradverteilung in den Systemen entstehen können.
In den folgenden Absätzen werden Literaturübersichten gegeben, die sich in die Klassen strategische, taktische, operative sowie die kombinierte taktische und operative Planung aufspalten.

Strategische Planung

Der strategischen Fragestellung widmen sich Lin et al. [45]. Sie schlagen einen Modellierungsansatz vor, der die Entscheidung über die Anzahl und Positionen der Stationen sowie die Netzwerkstruktur des Systems unterstützen soll. Dabei werden sowohl die betreiber- als auch die kundenseitigen Interessen simultan betrachtet. Auf der Seite des Betreibers stehen Kosten für den Bau der Stationen, Räder und Radwege. Der Servicelevel basiert auf der Abdeckung des Gebiets durch die Stationen und der Wahrscheinlichkeit, ein Fahrrad an der gewünschten Station zu erhalten. Zudem spielt auch der Fahrpreis eine Rolle. García-Palomares et al. [33] entwickeln ein geographisches Informationssystem, welches über die optimalen Positionen für Bike-Sharing-Stationen entscheiden, deren Charakteristik bestimmen sowie die Erreichbarkeit für die Kunden bewerten kann. Die Vorgehensweise erfolgt auf vier Stufen und wird am Beispiel vom Stadtzentrum Madrid angewandt. Zunächst wird die potentielle Kundennachfrage benötigt. Diese wird auf Basis von Einwohner und Arbeitnehmeranzahlen erzeugt. Es werden fünf Szenarien nach Anzahl der Stationen unterschieden. Mittels zwei Location-Allocation-Modellen (maximum coverage und minimum impedance) wird die Menge der Stationskandidaten und die zugehörige Lösung ermittelt. Diese wird dann entsprechend der Charakteristiken beschrieben. Im letzten Schritt erfolgt die Bewertung in Hinsicht auf die Abdeckung potentieller Zielorte. Martinez et al. [46] lösen das Problem des Systemdesigns mit einer Heuristik, die ein Mixed Integer Linear Program (MILP) beinhaltet. In einem iterativen Algorithmus werden gleichzeitig die Positionen der Stationen und die benötigte Repositionierungsflotte optimiert. Dabei wird auch ein Repositionierungskostenterm berücksichtigt, jedoch ist er keine Entscheidungsvariable. Die Zielfunktion maximiert den Gewinn, der sich aus den eingenommenen Fahrpreisen abzüglich der System- und Flottenkosten ergibt. Als Anwendungsbeispiel dient Lissabon.

Taktische Planung

Zu den rein taktischen Literaturbeiträgen gehört unter anderen der Beitrag von Shu et al. [73]. Mit einem Netzwerk-Flussmodell zur Maximierung der Radflüsse im System beantworten sie die Frage nach der optimalen Fahrrad- und Stellplatzanzahl an den gegebenen Stationen. Zudem diskutieren sie den Wert von periodischen Repositionierungsmaßnahmen, nehmen diese aber nicht in das Ent-

scheidungsproblem auf. Einen weiteren Beitrag liefern Raviv und Kolka [65]. Mit einem Closed-Loop IP modellieren sie das Füllstandsproblem. Die Zielfunktion beinhaltet selbst definierte Strafterme für nicht erfüllten Service bei Entleihung oder Rückgabe der Räder. Dabei betrachten sie allerdings die Stationen eines BSS einzeln und vernachlässigen die Interdependenzen. Cepolina und Farina [12] betrachten sogenannte Personal Intelligent City Accessible Vehicle (PICAV). Ähnlich wie BSS bieten PICAVs kleine elektrisch betriebene Fahrzeuge flexibel und kurzfristig zur Nutzung an. In ihrem Paper wird in einem Simulated Annealing-Ansatz über die Flottengröße und Verteilung der Fahrzeuge entschieden. Minimiert wird dabei eine Kombination aus Kosten basierend auf Kundenwartezeiten und für die Fahrzeugflotte.

Operative Planung

Beiträge zur rein operativen Planung betrachten meist das statische Problem, welches Repositionierungsaktivitäten in der Nacht annimmt und Kundenbewegungen währenddessen ausschließt. Eine Inputvariable ist dabei der Zielfüllstand an den Stationen bzw. gegebene Nachfragemengen an Fahrrädern und Stellplätzen. Häufig kommt eine Variante des Pickup-and-Delivery-Problems (PDP), welches sich gut zur Modellierung eignet, zum Einsatz.
Einen mathematischen Einblick in Herleitung und Komplexität bieten Benchimol et al. [6]. Mit einer Kombination des PDP und des Swapping Problems (SP) modellieren sie das statische Repositionierungsproblem mit einem Transportfahrzeug. Die Ergebnisse des Papers beinhalten untere Schranken und approximative Algorithmen. Chemla et al. [13] schlagen für das statische PDP mit einem Fahrzeug einen Branch-and-Cut-Algorithmus vor. Eine obere Schranke erhalten sie mittels Tabusuche. Di Gaspero et al. [24] stellen einen Ansatz kombiniert aus Constraint Programming (CP) und Ant Colony Optimization (ACO) vor. Sie minimieren die gewichtete Summe aus der Abweichung der Stationsfüllstände von einem Zielfüllstand und der Fahrtzeit für die Repositionierungen. Sie erweitern diesen Ansatz durch eine Large Neighborhood Search (LNS) [25] und liefern eine nochmals überarbeitete Version, die das Verfahren durch Branching-Strategien auf vielversprechende Lösungen konzentriert [26]. Ebenfalls der statischen Variante des Repositionierungsproblems widmen sich Rainer-Harbach et al. bzw. Raidl et al. in einer Serie von Veröffentlichungen. Zunächst entwickeln sie einen Variable Neighborhood Search (VNS) Ansatz, um Transporttouren zu generieren [63]. Über die

Ladeoperationen wird unabhängig davon unter anderem mit Greedy-Verfahren effektiv entschieden. Diese Entscheidung über die Ladeoperationen verbessern Raidl et al. in [61] signifikant. Die dritte Veröffentlichung [64] liefert eine bessere Konstruktionsheuristik, die sich der Weitsicht der Pilot-Methode bedient (siehe Voß et al. [81]). Ho et al. [39] senken die Komplexität des Problems, indem sie die Stationen eindeutig einer Pickup- oder Deliveryfunktion zuordnen. Mit einer iterativen Tabusuche minimieren sie dann die Strafkosten, die durch Fehlmengen an den Stationen entstehen. Dell'Amico et al. [22] formulieren vier unterschiedliche gemischt ganzzahlige lineare Programme (MILP) aufbauend auf Varianten des TSP, des one-commodity Pickup-and-Delivery TSP (1-PDTSP) und des VRP. Mit einem Branch-and-Cut-Verfahren minimieren sie die Transportkosten unter der Bedingung, dass die kundenseitige Differenz zwischen Rad- und Stellplatznachfrage ausgeglichen wird. Für Systeme mit bis zu 50 Stationen liefert der Algorithmus für verschiedene Benchmark-Instanzen in unter einer Stunde eine Lösung. Bei größeren Systemen steigen Rechenzeit und Abstände der ermittelten Lösung zu den unteren Schranken gewaltig an.

Kombinierte taktische und operative Planung

Wenige Beiträge heben sich von den übrigen statischen Ansätzen dadurch ab, dass sie sowohl über die Füllstände als auch über das Vehicle Routing entscheiden. Schuijbroek et al. [71] ermitteln zunächst Füllstandsintervalle, indem sie Markovketten modellieren. Diese werden als Servicebedingungen im weiteren Cluster-First-Route-Second-Verfahren verwendet. Das in Polynomialzeit lösbare Clusterproblem sichert die Einhaltung der Servicebedingungen ab, während es geschätzte Routingkosten mit einbezieht. Minimiert wird die maximale Tourlänge über alle Transportfahrzeuge. Raviv et al. [66] präsentieren zwei MILP zur Modellierung des statischen Repositionierungsproblems. Entschieden wird über die Anzahl der auf- und abzuladenden Räder sowie das Routing der Transportfahrzeuge. Die Zielfunktion minimiert eine Kombination aus Strafkosten und operativen Kosten.
Das dynamische Repositionierungsproblem wird erst in den letzten Jahren intensiver in der Literatur diskutiert. Hier wird angenommen, dass die Repositionierungen im laufenden Betrieb stattfinden. Auffällig ist, dass in den meisten untersuchten Beiträgen der Zusammenhang zwischen der taktischen und operativen Planung beachtet wird. Die Entscheidung über die zu repositionierenden Mengen fließt als Entscheidungsvariable in die Modelle ein. Dies passt die Modelle besser an das

reale Repositionierungsproblem an, macht sie aber zugleich komplexer.
In zwei aufeinanderfolgenden Veröffentlichungen entwickeln Caggiani und Ottomanelli [10] bzw. Caggiani et al. [11] ein Entscheidungsunterstützungssystem (DSS) und ein Micro-Simulationsmodell zur Bestimmung der optimalen Repositionierungsanzahlen und der Repositionierungsrouten sowie den Repositionierungsintervallen. Dabei berücksichtigen sie die dynamische Kundennachfrage. Als Anwendungsbeispiel können sie das Problem allerdings nur für ein künstliches System mit fünf Stationen lösen. Contardo et al. [19] modellieren das Repositionierungsproblem als Variante des one commodity PDP, in der auch über die Anzahl der zu transportierenden Räder entschieden wird. Sie schlagen eine Kombination aus Benders- [7] und Dantzig-Wolfe-Dekomposition [21] vor, um effektiv untere und obere Schranken für die Lösung zu erhalten. Die experimentellen Resultate sind allerdings noch weit von der optimalen Lösung entfernt, sodass eine weitere Entwicklung von Algorithmen als notwendig erachtet wird. Kloimüllner et al. [43] erweitern die vorhergehenden Beiträge von Raidl et al. und Rainer-Harbach et al. ([63], [61], [64]) zum dynamischen Fall, indem sie die Nachfrage und unerfüllte Nachfrage über den Tag betrachten. Sie minimieren eine zusammengesetzte Zielfunktion aus unerfülltem Service, Verfehlen der Zielfüllstände, Anzahl der repositionierten Räder und der Transportzeit. Dieser Beitrag ist allerdings in dieser Kategorie der einzige, welcher vorgegebene Zielfüllstände benötigt und somit eine separate Betrachtung der taktischen Ebene vorsieht.
Nach der Einordnung der einzelnen Problemstellungen der Forschungsbeiträge folgt die Hinführung zum Inhalt der vorliegenden Arbeit sowie dessen Abgrenzung zur bisher erschienenen Literatur. Im Kapitel 2.1.5 erfolgt die verbale Vorstellung der Problemstellung.
Inhalt dieser Arbeit soll die simultane taktische und operative Planung des dynamischen und stochastischen Repositionierungsproblems in BSS sein. Daher treffen wir zunächst die Annahme, dass das physische System, sprich die Stationen an den entsprechenden Orten mit zugehöriger Stellplatzanzahl, schon existiert. Die simultane Betrachtung der taktischen und operativen Ebene wird empfohlen, da sie stark voneinander abhängen. Eine optimale taktische Planung kann angemessene operative Nachteile oder Unzulässigkeit mit sich bringen und umgekehrt. Die statische Problemmodellierung wird ausgeschlossen, da die flexible und kurzzeitige Nutzung durch die Kunden eine erhebliche Dynamik mit sich bringt. Nur nächtlich zu repositionieren ist in einem gut laufenden BSS nicht ausreichend.
Der Forschungsbereich wird folglich auf die dynamischen Beiträge des Abschnitts

über *kombinierte taktische und operative Planung* eingegrenzt. Contardo et al. [19] nehmen eine deterministische Nachfrage an. Dies ist bei einem BSS, in dem Kunden unter anderem abhängig von Wetter, persönlichen Launen, spontanen Terminen etc. agieren, nicht empfehlenswert. Kloimüllner et al. [43] ermitteln die sogenannte kumulative erwartete Nachfrage $\mu_v(t)$ abhängig von der Zeit t als Differenz von erwarteten Zu- und Abflüssen an Rädern vom Startzeitpunkt der Repositionierungsaktion bis t. Diese Annahme lässt allerdings außer Acht, dass je nach Reihenfolge des Auftretens von Entleihungen oder Rückgaben trotzdem die Kapazitätsgrenzen erreicht werden könnten. Caggiani und Ottomanelli [11] sagen die Nachfrage mittels neuronaler Netze voraus. Diese haben den Vorteil, direkt aus beobachteten Daten lernen zu können. Jedoch kann deren Modell keine Instanzen von realer Größe handhaben.
Folglich kann keine der bisher in der Literatur erschienenen Arbeiten alle unsere Anforderungen erfüllen.

2.1.5. Problemstellung der vorliegenden Arbeit

Die Forschungslücke, welche mit dieser Arbeit geschlossen werden soll, ist also folgende:
Gesucht sind ein Modell, welches das Repositionierungsproblem für reale Systemgrößen abbilden kann und ein Verfahren, welches das Problem simultan auf taktischer und operativer Ebene löst. Dabei sollen sowohl die stochastischen als auch die dynamischen Aspekte des Problems beachtet werden. Die Idee und erste Ansätze sowie vorhergehende Versionen der Optimierung und der Simulation wurden von Ricker et al. [68] und Ricker und Mattfeld [67] veröffentlicht. Seither wurden Weiterentwicklungen an allen Bestandteilen des Ansatzes vorgenommen, die grundlegende Idee bleibt jedoch bestehen. Vorgeschlagen wird eine Modellierung des Problems als stochastisches und dynamisches Inventory Routing Problem (IRP). Da dieses $\mathcal{NP}$-schwere Problem nicht in annehmbarer Zeit lösbar ist, erfolgt eine Dekomposition (siehe Kapitel 2.2) des Gesamtproblems in einen taktischen und einen operativen Teil. In der taktischen Planung wird unter Antizipation der operativen Ergebnisse mittels Approximativer Dynamischer Programmierung (siehe Kapitel 2.3) über das Inventory Problem (IP) entschieden. Das Routing erfolgt heuristisch. Die Auswirkungen auf das BSS werden simulativ gemessen und fließen in den Planungskreislauf ein. Die Problemformulierung sowie die mathematische Sicht auf die Vorgehensweise des vorgeschlagenen Verfahrens erhält der Leser in Kapitel 3. Eine Ebenensicht auf das Verfahren mit Augenmerk auf Flüsse und Ver-

arbeitung von Informationen wird in Kapitel 4 beschrieben. Die Umsetzung des vorgeschlagenen Verfahrens wird im Kapitel 5 erläutert, bevor im Kapitel 6 die Ergebnisse der Experimente zusammengetragen werden.

2.2. Dekompositionsverfahren für Optimierungsprobleme

Die Idee, ein komplexes Problem in kleine Teile zu brechen, um diese einzeln zu betrachten und später zu rekombinieren, ist in der Literatur schon früh zu finden (siehe Polya [57]). Das Prinzip besteht darin, das Ausgangsproblem in Master- und Subprobleme zu teilen, die jede für sich effizient gelöst werden können. Iterativ kann man dann die beiden Problemgruppen bearbeiten und gegenseitig mit Informationen der jeweils anderen Gruppe versorgen [76]. Prominente Beispiele sind die bereits im Literaturkapitel 2.1.4 erwähnten Benders'- und Dantzig-Wolfe-Dekomposition. Während Benders [7] Variablen aus dem Masterproblem eliminiert, um sie als Schnittebenen wieder einzufügen, wird bei Dantzig-Wolfe [21] Spaltengenerierung für die Wiedereinfügung genutzt. Rothlauf diskutiert in Kapitel 2.4.3 seines Buchs [69] die Dekomponierbarkeit von Optimierungsproblemen. Sind die Entscheidungsvariablen in unabhängige Mengen aufteilbar, so ist sie hoch. Gibt es viele Interaktionen zwischen den Gruppen von Variablen, so ist sie niedrig. Je höher die Dekomponierbarkeit von einem Problem ist, desto besser ist die Performance eines Dekompositions- und Rekombinierungsansatzes zu erwarten.
Der Dekompositionsansatz, welcher hier im Falle des Bike-Sharings Anwendung findet, wird in Kapitel 5 vorgestellt. Dabei wird der Tagesablauf im BSS als Markovprozess modelliert. Für die Optimierung des taktischen Teils kommt ein Ansatz mittels Approximativer Dynamischer Programmierung zum Einsatz. Die zugehörigen Grundlagen werden im folgenden Kapitel 2.3 gelegt.

2.3. Approximative Dynamische Programmierung

Dieses Kapitel führt in die Approximative Dynamische Programmierung (ADP) ein. Dazu wird in Kapitel 2.3.1 zunächst der dynamische Entscheidungsprozess allgemein erläutert. Antizipation wird darauf folgend in Kapitel 2.3.2 eingeführt.

2.3.1. Dynamische Entscheidungsprozesse

Bellman [3] definiert den dynamischen Entscheidungsprozess als ein System, welches zu jeder Zeit $t \in [0, T]$ einen Zustand S_t hat, welcher durch eine Menge von Zustandsvariablen beschrieben werden kann. Zu bestimmten Entscheidungszeitpunkten k mit $k = 1, ..., K$ werden Entscheidungen d_k aus dem Entscheidungsraum $\mathcal{D}$ über die Transformation des Systemzustands getroffen. Diese sollen die Folgezustände und Folgeentscheidungen so beeinflussen, dass eine Zielfunktion, abhängig von den Zustandsparametern des finalen Zustands, optimiert wird. Eine Folge $\pi \in \Pi$ von Entscheidungen heißt *Politik*. Optimiert sie die entsprechende Zielfunktion, so nennen wir sie *optimale Politik*. Ein spezieller Entscheidungsprozess ist der *Markovprozess* (MDP) [4]. Dessen wichtigste Eigenschaft ist, dass eine Entscheidung und deren Kosten bzw. Ertrag c_k nicht von den Vorgängern von S_k abhängen. Ein Zustandsübergang $S_k \longrightarrow S_{k+1}$ kann je nach Problemstellung entweder nur von der Entscheidung $d \in \mathcal{D}$ herbeigeführt sein ($S_k \xrightarrow{d_k} S_{k+1}$) oder zusätzlich von äußeren Einflüssen ω deterministischer oder stochastischer Art ($S_k \xrightarrow{d_k,\omega_k} S_{k+1}$). Im Folgenden beschränken wir uns auf die Betrachtung von stochastischen Prozessen, welche unsicheren Einflüssen unterliegen.
Um ein dynamisches Programm für den dynamischen Prozess zu definieren nutzt Powell [58] folgende Elemente:

- **Zustandsvariable** - Informationen zur Beschreibung des Systems und zur Entscheidungsfindung
- **Entscheidungsvariable** - Entscheidung d_k über den Eingriff in das System
- **äußerer Einfluss** - neue Information ω_k, die in jeder Periode bekannt werden
- **Zustandsübergang** - Beschreibung der Entwicklung des Systems von S_k nach S_{k+1} unter Berücksichtigung von Entscheidung d_k und des äußeren Einflusses zwischen k und $k + 1$
- **Kosten- bzw. Ertragsfunktion** - Legt die entstandenen Kosten oder Erträge c_k im entsprechenden Zeitintervall fest
- **Zielfunktion** - Formaler Ausdruck des Optimierungsziels (Kostenminimierung oder Ertragsmaximierung)

Formal kann obiger Zustandsübergang $S_k \longrightarrow S_{k+1}$ durch folgende Übergangsfunktion S^M ausgedrückt werden:

$$S_{k+1} = S^M\left(S_k, d_k, \omega_k\right). \tag{2.1}$$

Dabei können Kosten oder Erträge entstehen, die abhängig von Zustand, Entscheidung, äußerem Einfluss oder einer Kombination aus den Elementen sind. Möglich wären unter anderem folgende Beispiele.

$$c_k\left(S_k, d_k\right) \tag{2.2}$$

$$c_k\left(S_k, d_k, \omega_k\right) \tag{2.3}$$

In einem deterministischen Problemfall mit einem einzigen Kostenterm wäre eine Zielfunktion zur Kostenminimierung durch

$$\min_{(d_k)_{k=1}^K} \sum_{k=1}^{K} c_k\left(S_k, d_k\right) \tag{2.4}$$

formulierbar.
Liegen allerdings stochastische Einflüsse vor, wird S_t zu einer Zufallsgröße. In diesem Fall benötigen wir eine entsprechende Entscheidungsregel, die uns eine Politik π vorgibt. Sei $d^\pi(S_k)$ die Entscheidungsregel. Dann lautet die Zielfunktion zur Bestimmung der optimalen Politik

$$\min_{\pi \in \Pi} \mathbb{E} \sum_{k=1}^{K} c_k\left(S_k, d^\pi(S_k)\right). \tag{2.5}$$

In beiden Fällen ist Weitsicht hilfreich. Das heißt, zu jedem Entscheidungszeitpunkt k sind die Auswirkungen von d_k auf alle folgenden Zustände und Entscheidungen zu berücksichtigen. Diese Weitsicht wird Antizipation genannt und wird im folgenden Kapitel 2.3.2 eingeführt.

2.3.2. Antizipation

Das Nomen *Antizipation* bedeutet wörtlich Vorgriff. Es stammt aus dem Lateinischen und lässt sich von ante (vor) und capere (nehmen) ableiten [29]. Im Kontext des dynamischen Entscheidungsprozesses soll zu den Entscheidungszeitpunkten k der restliche Prozessverlauf antizipiert werden, um die Entscheidung unter Einbe-

ziehung zukünftiger Ereignisse zu optimieren. Dem entgegen stehen reaktive Entscheidungen, welche nur auf Basis vorliegender Informationen getroffen werden. Powell hat experimentell gezeigt, dass antizipierende Strategien in stochastischen und dynamischen Problemen (zum Beispiel in der Tourenplanung) bessere Lösungen erzeugen als reaktive [59]. In Anlehnung an Meisel [49] wird in diesem Abschnitt gezeigt, inwiefern Antizipation die Entscheidungsfindung in dynamischen Prozessen verbessert.
Eine antizipierende Entscheidung d_k kann zwei verschiedene Einflüsse auf das Ziel des Entscheiders haben. Zum einen können direkte Kosten $c_k\left(S_k, d_k\right)$ entstehen. Zum anderen hängt der Folgezustand S_{k+1} von ihr ab. Die Abhängigkeit zieht sich durch den gesamten Prozess und wirkt sich auf zukünftige Zustände, Entscheidungen und Kosten aus. Dabei soll die Entscheidung so gewählt werden, dass das optimale Gesamtergebnis bezüglich der direkten und zukünftigen Kosten erzielt wird. Bei dieser Entscheidung hilft die Definition der Wertefunktion $V(S_k)$ eines Zustandes S_k, der jeweils der entsprechenden Teilsumme

$$\min_{\pi \in \Pi} \mathbb{E} \sum_{\kappa=k}^{K} c_\kappa \left(S_\kappa, d^\pi(S_\kappa)\right). \tag{2.6}$$

aus Formel 2.5 entspricht. Die Definition lautet wie folgt (siehe Powell [58]):
Der Wert $V(S_k)$ des Zustands S_k lässt sich rekursiv aus den Kosten und dem erwarteten Wert des Folgezustands bestimmen.

$$V(S_k) = \min_{d_k \in \mathcal{D}} \left(c_k(S_k, d_k) + \mathbb{E}\left\{\gamma V(S_{k+1}) | S_k\right\}\right), \tag{2.7}$$

wobei $\gamma \leq 1$ als Diskontierungsfaktor eingefügt werden kann, welcher zukünftige Werte gegenüber dem aktuellen geringer wertet.
Die optimale Entscheidung d_k^* errechnet sich folglich aus

$$d_k^* = \underset{d_k \in \mathcal{D}}{\operatorname{argmin}} \left(c_k(S_k, d_k) + \mathbb{E}\left\{\gamma V(S_{k+1}) | S_k\right\}\right). \tag{2.8}$$

Das zugehörige Optimierungsproblem im Zustand S_k lautet

$$\begin{aligned} &\min \left(c_k(S_k, d_k) + \mathbb{E}\left\{\gamma V(S_{k+1}) | S_k\right\}\right) \\ &\text{mit } d_k \in \mathcal{D}. \end{aligned} \tag{2.9}$$

2.3.2.1. Perfekte Antizipation

Um perfekt zu antizipieren, ist die Lösung von Optimierungsproblem 2.9 notwendig. Dies ist nur mit der Kenntnis aller Werte $V(S_{k+1})$ möglich. Die dynamische Programmierung bietet grundlegende Methoden, die dieses Optimierungsproblem lösen können. Jedoch scheitert deren praktische Anwendung aus drei Gründen [58]:

- Die *Zustandsraumgröße* ist eine wichtige Einflussgröße für den Rechenaufwand der Methoden der dynamischen Programmierung. Sie ist abhängig von der Anzahl K der Entscheidungszeitpunkte sowie der Anzahl an Attributen, die zur Beschreibung des Systemzustandes S_k benötigt werden.

- Die *Erwartungswerte*, die für jeden Zustand berechnet werden müssen, machen das Problem des großen Zustandsraumes noch gravierender. Da die beteiligten Größen nicht notwendigerweise unabhängig sind, können die Berechnungen eine große Anzahl von verschachtelten Summen enthalten.

- Die Größe des *Entscheidungsraums* legt fest, wie häufig die Erwartungswertberechnungen stattfinden müssen. Sie wird durch die Anzahl der Attribute zur Beschreibung der Entscheidung sowie deren mögliche Ausprägungen bestimmt.

Folglich explodiert die Komplexität schon bei relativ kleinen Probleminstanzen und führt zu Rechenzeit- sowie Speicherkapazitätsproblemen. Es ist also nicht ohne weiteres möglich, mit perfekter Antizipation zu arbeiten, wenn man stochastische und dynamische Probleme von realer Größe betrachtet. Will man nicht auf Antizipation verzichten, so bietet die approximative Antizipation eine Alternative, die unter Verlust der Genauigkeit die Komplexität verringert.

2.3.2.2. Approximative Antizipation

Die Idee der ADP existiert schon recht lang. 1959 veröffentlichten Bellman und Dreyfus die ersten Anfänge [5], wenige Jahre nach Bellmanns wichtigen Papern zur dynamischen Programmierung [3]. In den Bereichen der Kontrolltheorie [82] und der künstlichen Intelligenz [41] existieren Weiterentwicklungen und Anwendungsbeispiele zu ADP. Eine neuere interdisziplinäre Übersicht stammt von Si et al. [74].
Powell stellt in seinem Buch [58] ökonomische Problemstellungen in den Vordergrund. Er hebt hervor, dass ADP andere Forschungsfelder als Spezialfälle umfasst.

Dies führt zu einem breiten Anwendungsbereich der vorgestellten Algorithmen. Betrachten wir noch einmal die Formeln 2.7, 2.8 und das zugehörige Optimierungsproblem 2.9. Um approximativ zu optimieren, ist eine Abschätzung des Erwartungswertes notwendig:

$$\overline{V}(S_{k+1}) \approx \mathbb{E}\left\{V(S_{k+1})|S_k\right\}.$$

Ein Lernfortschritt kann dabei durch mehrfaches Durchlaufen des Zeithorizontes erzielt werden. Während der Durchläufe $m = 0, ..., M$ (sog. Trajektorien) kann der jeweilige geschätzte Wert $\overline{V}^m(S_k)$ der Zustände S_k aktualisiert werden. Dadurch wird die Aufzählung aller Wahrscheinlichkeitsausgänge umgangen. In der Regel simuliert man das System und kann so, durch die Aktualisierungen der Werte, die entsprechende Performance der einzelnen Zustände bzw. Entscheidungen ablesen. Je häufiger Zustände erreicht werden, desto fundierter wird das erlernte Wissen. Anhand dieses Wissens ist dann die optimale Entscheidung über den Eingriff in das System zu treffen. Das entsprechende Vorgehen mit einem Vorwärts-Programmierungs-Ansatz wird in Algorithmus 1 veranschaulicht.
α_m ist dabei die sogenannte Schrittweite. Sie hat eine glättende Aufgabe und nimmt in der Regel Werte im Bereich $(0, 1)$ an. Diese können fix gesetzt sein, oder einer Funktion wie z.B. $\alpha_m = \frac{1}{m+1}$ folgen. So können gemittelte Werte aus stochastisch verrauschten Daten $\hat{v}_k^m$ ermittelt werden. Zurückgegeben wird die sogenannte lookup-table, eine Tabelle mit den ermittelten Werten für jeden Zustand. Diese Tabelle kann für zukünftige Entscheidungen in Testsettings herangezogen werden und wird im Folgenden mit *Wertetabelle* bezeichnet.
Bei der Entwicklung von ADP-Algorithmen liegt eine große Schwierigkeit in der Berechnung oder Abschätzung von Erwartungswerten innerhalb der Minimierungsoperation (siehe Formel 2.10). Um dies zu umgehen, kann ein künstlicher *Post-Decision-Zustand* (PDS) $S_k^{d_k}$ eingeführt werden. Sei also $S_k^{d_k}$ der deterministische Zustand, der unmittelbar der Entscheidung d_k folgt und ohne den äußeren Einfluss ω_k eintreten würde. Im Zustandsübergang ist nun d_k von ω_k abgrenzbar:

$$S_k \xrightarrow{d_k} S_k^{d_k} \xrightarrow{\omega_k} S_{k+1} \tag{2.11}$$

Der Wert $V(S_{k-1}^{d_{k-1}})$ ergibt sich nun aus

$$V(S_{k-1}^{d_{k-1}}) = \mathbb{E}\left\{\min_{d_k \in \mathcal{D}}\left(c_k(S_k, d_k) + \gamma V(S_k^{d_k})|S_{k-1}^{d_{k-1}}\right)\right\}. \tag{2.12}$$

Algorithmus 1: Genereller ADP-Algorithmus

Initialisierung:
Initialisiere $V^0(S_k)$, $\forall S_k$.
Setze M.
$m = 1$
Initialisiere S_0.
while $m \leq M$ **do**
  Generiere Zufallsnachfragen ω^m.
  for $k = 1, ..., K$ **do**
    Wähle ein zufälliges Sample $\hat{\Omega}_{k+1}^m$ aus den möglichen Realisierungen des zufälligen Einflusses zwischen k und $k+1$.
    Löse

$$\hat{v}_k^m = \min_{d_k \in \mathcal{D}} \left(c_k(S_k, d_k) + \sum_{\hat{\omega} \in \hat{\Omega}_{k+1}^m} \left(p_{k+1}(\hat{\omega}) \cdot \overline{V}^{m-1}(S_{k+1}) \right) \right) \tag{2.10}$$

    mit $d_k^{m*} = \underset{d_k \in \mathcal{D}}{\operatorname{argmin}} \left(c_k(S_k, d_k) + \sum_{\hat{\omega} \in \hat{\Omega}_{k+1}^m} \left(p_{k+1}(\hat{\omega}) \cdot \overline{V}^{m-1}(S_{k+1}) \right) \right)$.

    if $k > 1$ **then**
      Aktualisiere $\overline{V}^{m-1}(S_{k+1})$:
      $\overline{V}^m(S_k) = (1 - \alpha_{m-1})\overline{V}^{m-1}(S_k) + \alpha_{m-1}\hat{v}_k^m$
    end
    Gehe zum nächsten Zustand über:
    $S_k^m \xrightarrow{d_k^{m*}, \omega_k^m} S_{k+1}^m$
  end
  Erhöhe m.
end
return $\overline{V}^M(S_k)$, $\forall S_k$

Es ist sofort ersichtlich, dass nun der Erwartungswert nach außen gewandert ist. Die scheinbare Schwierigkeit, ein Optimierungsproblem innerhalb eines Erwartungswert lösen zu müssen, stellt sich als programmiertechnischer Vorteil heraus. Sei $\overline{V}\left(S_k^{d_k}\right)$ eine Abschätzung des Wertes des PDS $S_k^{d_k}$. Angenommen, wir befinden uns in Iteration m eines ADP-Algorithmus. Dann lässt sich die Entscheidung wie folgt auswählen:

$$\hat{v}_k^m = \min_{d_k \in \mathcal{D}} \left(c_k(S_k, d_k) + \gamma \overline{V}\left(S_k^{d_k}\right) \right). \tag{2.13}$$

Der wichtige Vorteil ist, dass dieses Minimierungsproblem jetzt deterministisch ist. Ein weiterer Vorteil und zusätzlich eine Effizienzsteigerung bietet sich dadurch, dass $\hat{v}_k^m$ eine Realisierung mit doppelter Information ist. Zum einen kann es zur Bewertung von $S_k^{d_k}$ und zum anderen zur Bewertung von d_k genutzt werden, da d_k uns in den entsprechenden PDS geführt hat. Folglich können wir die Werte $\overline{V}\left(S_{k-1}^{d_{k-1}}\right)$ wie folgt aktualisieren:

$$\overline{V}^m(S_{k-1}^{d_{k-1}}) = (1 - \alpha_{m-1})\overline{V}^{m-1}(S_{k-1}^{d_{k-1}}) + \alpha_{m-1}\hat{v}_k^m. \tag{2.14}$$

Die Betrachtung des PDS vereinfacht oft das Erreichen einer guten Approximation [58]. Mit dieser Umstellung können wir aus Algorithmus 1 den verbesserten Algorithmus 2 ableiten.

Durch die Nutzung des PDS umgehen wir die Schätzung des Erwartungswertes innerhalb der Optimierungsproblems. Außerdem ist die Struktur der Werte $\overline{V}(S_k^{d_k})$ ohne Erwartungswert direkt kontrollierbar und so je nach Notwendigkeit anpassbar.

Trotz des eleganten Vorgehens bleiben die zwei anderen Komplexitätsherausforderungen des Zustands- und Entscheidungsraums bestehen. Ein zu großer Zustandsraum brächte ebenso große Wertetabellen mit sich. Davon abgesehen, dass diese zu einem Speicherproblem führen können, wäre die Lernperformance geschwächt, da der einzelne Zustand weniger oft besucht würde. Dieses Problem löst eine Aggregation der Zustände. Dies kann auf verschiedenste Arten und Weisen geschehen. Mögliche Werkzeuge sind z.B. Clusterings zum Zusammenfassen der Zustände zu Gruppen. Die Größe des Entscheidungsraums bestimmt die Anzahl der Rechenoperationen in Formel 2.15. Je mehr Optionen es gibt, desto umfangreicher ist die Suche nach dem Minimum. Auch hier kann es helfen, Entscheidungen zu bündeln, oder durch Dekomposition die Entscheidungsvariablen auf Teilprobleme zu beziehen und so zu vereinfachen.

Algorithmus 2: ADP-Algorithmus mit Post-Decision-Zustand

Initialisierung:
Initialisiere $V^0(S_k^{d_k})$, $\forall S_k^{d_k}$.
Setze M.
$m = 1$
Initialisiere S_0.
while $m \leq M$ **do**
    Generiere Zufallsnachfragen ω^m.
    for $k = 1, ..., K$ **do**
        Löse

$$\hat{v}_k^m = \min_{d_k \in \mathcal{D}} \left(c_k(S_k, d_k) + \gamma \overline{V} \left(S_k^{d_k} \right) \right) \tag{2.15}$$

        mit $d_k^{m*} = \underset{d_k \in \mathcal{D}}{\operatorname{argmin}} \left(c_k(S_k, d_k) + \gamma \overline{V} \left(S_k^{d_k} \right) \right)$.
        if $k > 1$ **then**
            Aktualisiere $\overline{V}^{d_k m-1}(S_{k-1}^{d_k})$:
            $\overline{V}^{d_k m}(S_{k-1}^{d_k}) = (1 - \alpha_{m-1}) \overline{V}^{d_k m-1}(S_{k-1}^{d_k}) + \alpha_{m-1} \hat{v}_k^m$
        end
        Gehe zum nächsten Post-Decision-Zustand über:
        $S_k^m \xrightarrow{d_k^{m*}} S_k^{d_k m}$
        Gehe zum nächsten Zustand über:
        $S_k^{d_k m} \xrightarrow{\omega_k^m} S_{k+1}^m$
    end
    Erhöhe m.
end
return $\overline{V}^M(S_k^{d_k})$, $\forall S_k^{d_k}$

Der Einsatz dieser Algorithmen und Aggregationswerkzeuge wird in den folgenden Kapiteln 3 aus mathematischer und 4 aus methodischer Sicht erläutert. Die Umsetzung erfolgt in Kapitel 5.

3. Problemformulierung und -einordnung

Dieses Kapitel modelliert die Problemstellung aus Kapitel 2.1.5 mathematisch. Unter Anwendung der Notation aus Kapitel 2.3 wird die Komplexität des Problems dargelegt. In Kapitel 3.3 und 3.4 folgen die Maßnahmen zur Reduktion der Komplexität mittels Dekomposition und Aggregation.

3.1. Mathematische Formulierung

In Kapitel 2.1.3 wurde zusammengetragen, welche Teile des Repositionierungsproblems durch welche Modelle der Literatur beschrieben werden können. Da taktische und operative Maßnahmen kombiniert betrachtet werden, handelt es sich bei dem in Kapitel 2.1.5 verbal formulierten Problem um eine Form des Inventory Routing Problems (IRP). Coelho et al. [15] geben in ihrem Paper eine umfassende Übersicht über Varianten, Modelle und Algorithmen des IRP aus den letzten 30 Jahren. Bevor zur Definition des IRP übergegangen wird, folgt zunächst ein Kapitel zur Klassifikation von mathematischen Modellen.

3.1.1. Klassifikation von Modellen

Bei der Charakterisierung von Modellen sind für die folgenden Kapitel zwei Eigenschaften von besonderer Relevanz (siehe [27] oder [42]).

- Informationsgrad über die Daten
- Zeitliche Entwicklung von Modellparametern

Im Hinblick auf den Informationsgrad werden *deterministische* und *stochastische* Modelle unterschieden. Während deterministische Modelle nur feste Modellparameter enthalten, also alle Informationen sicher sind, nimmt man in stochastischen Modellen bestimmte Verteilungen an bzw. schätzt mehrere mögliche Werte sowie

deren Wahrscheinlichkeiten. Für deterministische Modelle lassen sich mit geeigneten Verfahren optimale Lösungen finden, für stochastische sind nur Aussagen über wahrscheinliche Lösungsgüten möglich.
Bei der zeitlichen Entwicklung von Modellparametern grenzt man *statische* von *dynamischen* Modellen ab. Statische Modelle beinhalten im Planungszeitraum unveränderliche Parameter. Parameter in dynamischen Modellen können hingegen zeitlichen Schwankungen unterliegen. Statische Modelle können sinnvolle Ergebnisse bei kurzen Planungshorizonten liefern, wenn eine Entscheidung keinen Einfluss auf Folgeperioden hat. Da dies selten der Fall ist, sollten Entscheidungen in dynamischen Problemen auch basierend auf dynamischen Modellen getroffen werden.

3.1.2. Das Inventory Routing Problem

Das grundlegende IRP-Modell ist auf dem Graphen $\mathcal{G}(\mathcal{V}, \mathcal{E})$ mit der Knotenmenge $\mathcal{V} = \{0, ..., n\}$ und der Kantenmenge $\mathcal{E} = \{(i, j) | i, j \in \mathcal{V} \text{ mit } i \neq j\}$ definiert. Knoten 0 bezeichnet dabei das Depot, die übrigen Knoten i Kundenstandorte. Für Depot sowie Kundenstandorte fallen Lagerhaltungskosten c_i^l pro Periode $t \in \{0, ..., T\}$ an. C_i bezeichnet die Lagerkapazität jedes Knoten i. Die Produktionsrate im Depot wird mit r^t bezeichnet und es soll die Annahme gelten, dass genug produziert wird, um allen Bedarf $\sum_{i=1}^{n} \sum_{t=0}^{T} d_i^t$ zu decken. Lagerbestände I_i^t am Knoten i zum Zeitpunkt t sind nichtnegativ. Eine Menge von Transportfahrzeugen $v \in \{1, ..., V\}$ mit Ladekapazität Q_v steht zur Verfügung. Die Kosten im Sinne der Fahrtzeit oder Benzinverbrauch (je nach Anwendungsfall) auf Kante (i, j) betragen c_{ij}^{tr}. Das Ziel ist, die Gesamtkosten zu minimieren, während der Bedarf eines jeden Kunden gedeckt wird. Eine Lösung des Problems legt fest, zu welcher Periode t welcher Kunde i von welchem Transporter v mit welcher Menge q_i^{vt} beliefert wird.
Das IRP gehört schon in dieser grundlegenden *statischen* und *deterministischen* Formulierung in die Klasse der $\mathcal{NP}$-schwierigen Probleme, da es das Vehicle Routing Problem (VRP) umfasst. Es gibt trotzdem einige Vorschläge für exakte Algorithmen in der Literatur. Archetti et al. [2] schlagen einen Branch-and-Cut-Algorithmus zur exakten Lösung des IRP mit einem Fahrzeug vor. Dieser Ansatz wird mit Kürzeste-Wege-Netzwerken und einer initialen oberen Schranke durch Solyalı und Süral [75] verbessert. Coelho et al. [16] haben neue Algorithmen eingeführt, die den Ansatz von Archetti et al. [2] auf den Fall mit mehreren Fahrzeugen erweitern. In den Formeln 3.1 ff. wird diese Version des IRP unter der Annahme,

dass die Lager der Kunden immer voll gefüllt werden, formuliert.

$$\min \left(\sum_{i=1}^{n} \sum_{t=0}^{T} c_i^l I_i^t + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i<j}}^{n} \sum_{v=1}^{V} \sum_{t=0}^{T} c_{ij}^{tr} x_{ij}^{vt} \right) \tag{3.1}$$

$$\text{u.d.N. } I_0^t = I_0^{t-1} + r^t - \sum_{v=1}^{V} \sum_{i=1}^{n} q_i^{vt}, \forall t \tag{3.2}$$

$$I_i^t = I_i^{t-1} + \sum_{v=1}^{V} q_i^{vt} - d_i^t, \forall i \in \mathcal{V} \backslash \{0\}, \forall t \tag{3.3}$$

$$I_i^t \geq 0, \forall i, \forall t \tag{3.4}$$

$$I_i^t \leq C_i, \forall i, \forall t \tag{3.5}$$

$$\sum_{v=1}^{V} q_i^{vt} \leq C_i - I_i^{t-1}, \forall i \in \mathcal{V} \backslash \{0\}, \forall t \tag{3.6}$$

$$q_i^{vt} \geq C_i y_i^{vt} - I_i^{t-1}, \forall i \in \mathcal{V} \backslash \{0\}, \forall v, \forall t \tag{3.7}$$

$$q_i^{vt} \leq C_i y_i^{vt}, \forall i \in \mathcal{V} \backslash \{0\}, \forall v, \forall t \tag{3.8}$$

$$\sum_{v=1}^{V} q_i^{vt} \leq Q_v y_0^{vt}, \forall v, \forall t \tag{3.9}$$

$$\sum_{\substack{j=1 \\ i<j}}^{n} x_{ij}^{vt} + \sum_{\substack{j=1 \\ i>j}}^{n} x_{ji}^{vt} = 2 y_i^{vt}, \forall i, \forall v, \forall t \tag{3.10}$$

$$\sum_{i \in \mathcal{S}} \sum_{\substack{j \in \mathcal{S} \\ i<j}} x_{ij}^{vt} \leq \sum_{i \in \mathcal{S}} y_i^{vt} - y_m^{vt}, \forall \mathcal{S} \subseteq \mathcal{V}, \forall v, \forall t, \forall m \in \mathcal{S} \tag{3.11}$$

$$q_i^{vt} \geq 0, \forall i \in \mathcal{V} \backslash \{0\}, \forall v, \forall t \tag{3.12}$$

$$x_{ij}^{vt} \in \{0,1\}, \forall i,j \in \mathcal{V}, \forall v, \forall t \tag{3.13}$$

$$y_i^{vt} \in \{0,1\}, \forall i \in \mathcal{V} \backslash \{0\}, \forall v, \forall t \tag{3.14}$$

Formel 3.1 ist die Zielfunktion, welche die Summe aus Lager- und Routingkosten minimiert. Die Entscheidungsvariablen sind x_{ij}^{vt}, welche beschreiben, ob die Kante von i nach j in t von Fahrzeug v befahren wird, und y_i^{vt} welche besagen, ob Station i von Fahrzeug v in t angefahren wird. 3.2 und 3.3 beschreiben die Bestandsübergänge in Depot- und Kundenlagern. 3.4 und 3.5 beschränken die Lagerbestände durch 0 und die jeweilige Kapazität C_i an den Stationen i. Die Restriktionen 3.6 bis 3.8 bewirken, dass die Liefermengen die Kapazitäten der Lager nicht überschreiten

aber sie vollständig füllen und dass nur geliefert werden kann, wenn das Fahrzeug den Kunden auch besucht. 3.9 stellt sicher, dass kein Fahrzeug zu viel lädt. Bedingung 3.10 sorgt für gleichen Zu- und Abfluss von Fahrzeugen an den Knoten und Bedingung 3.11 verhindert die Bildung von Subtouren. 3.12 bis 3.14 sind die Nichtnegativitätsbedingung für die Liefermengen sowie die Binärbedingungen für die Kantennutzung bzw. den Besuch der Stationen.

Die heuristischen Verfahren im Gebiet des IRPs haben eine offensichtliche Entwicklung durchlebt. Während die frühen Ansätze vereinfachte Problemstellungen betrachteten, können neuere Verfahren auch komplexe Probleme lösen. Erstere durchsuchten mit einfachen Nachbarschaftsoperatoren den Suchraum und dekomponierten das IRP in hierarchische Subprobleme. Beispielhaft ist hier der Beitrag von Dror und Levy [28] zu nennen, welcher Verbesserungsverfahren für das IRP vorschlägt. Neuere Verfahren bauen auf Metaheuristiken, welche lokale Suchen mit Strategien verbinden, die lokale Optima wieder verlassen können. Folgende Beiträge sind allgemeiner Natur und nicht auf bestimmte Problemstellungen bezogen. Das Buch „Design of Modern Heuristics“ von Rothlauf [69] beschreibt die grundlegenden Konzepte hinter verschiedenen Heuristiken und erläutert von Grund auf, wie ein Problem beschaffen sein muss, damit eine Heuristik effizient darauf zugeschnitten werden kann. Ein Handbuch über Metaheuristiken veröffentlichten Gendreau und Potvin 2003, eine Neuauflage erfolgte 2010 [34]. Es umfasst neben vielen Beiträgen zu verschiedensten Metaheuristiken auch ein Paper von Raidl et al., welches die Weiterentwicklung metaheuristischer Ansätze durch die Hybridisierung verschiedener Metaheuristiken beschreibt [62]. Hybride Metaheuristiken performen demnach bei bestimmten Problemstellungen besser als ihre jeweiligen Bestandteile.

Es wird jedoch in keiner dieser Arbeiten das dynamische und stochastische IRP (DSIRP) thematisiert. Diese beiden Eigenschaften machen das IRP ungleich schwieriger. Um der Modellierung des Repositionierungsproblems näher zu kommen, müssen sie aber mit einbezogen werden. Die mathematische Formulierung unterscheidet sich in der Hinsicht, dass die Nachfrage d_i^t nun eine Zufallsvariable ist. Zudem erweitert sich die Zielfunktion um einen Strafterm, der nicht erfüllte Nachfrage bestraft. Die Zielfunktion aus Formel 3.1 wird dann zu

$$\min \left(\sum_{i=1}^{n} \sum_{t=0}^{T} c_i^l I_i^t + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i<j}}^{n} \sum_{v=1}^{V} \sum_{t=0}^{T} c_{ij}^{tr} x_{ij}^{vt} + \sum_{i=1}^{n} \sum_{t=0}^{T} c^p l_i^t \right) \tag{3.15}$$

wobei c^p der Strafkostenterm und l_i^t die Fehlmenge an Knoten i zur Zeit t ist. Um keine negativen Bestände zu erhalten, nimmt man die Fehlmengen mit in die Bestandsgleichungen auf.

$$I_i^t = I_i^{t-1} + \sum_{v=1}^{V} q_i^{vt} - d_i^t + l_i^t, \forall i \in \mathcal{V}\backslash\{0\}, \forall t \tag{3.16}$$

Der dynamische Aspekt wird in der Literatur durch das mehrfache Lösen von statischen Modellen eingebaut. Dabei kommen Verfahren zum Einsatz, die die Möglichkeit haben, zukünftige Ereignisse abzuschätzen und in die Lösung einzubeziehen. Dies ist auch in den folgenden zwei Beiträgen der Fall. Bertazzi et al. [8] stellen eine Modellierung des DSIRP als dynamisches Programm vor. Dieses lösen sie heuristisch mit einem hybriden Rollout-Algorithmus. Die Idee dabei ist, mittels einer Heuristik, der sogenannten base policy (Basispolitik), die Kosten eine Periode im Voraus abzuschätzen. Der Algorithmus wurde auf Instanzen mit drei und sechs Perioden getestet und konnte sie mit bis zu 50 bzw. 25 Kunden lösen. Coelho et al. [17] vergleichen verschiedene Lösungsstrategien des DSIRP und binden diese in ein dynamisches rolling-horizon-Framework ein. In jeder Periode wird das statische Problem mit einer adaptiven Large-Neighborhood-Search (ALNS) gelöst. Sie nutzen außerdem Nachfragevorhersagen und Umladevorgänge und evaluieren deren Einfluss auf die Lösung. Dieses Verfahren wurde auf Instanzen mit bis zu 200 Kunden und 20 Perioden getestet. Beide Paper betrachten das DSIRP mit nur einem Fahrzeug.
Mit dem DSIRP sind wir der Modellierung des Repositionierungsproblems aus 2.1.5 zwar ein Stück entgegen gekommen. Stationen und Depot im BSS entsprechen Kunden und Depot im IRP, die Fahrzeugmenge und deren Routingbedingungen können auch weitestgehend übernommen werden. Allerdings gibt es noch einige gravierende Unterschiede. In Kapitel 3.1.3 werden diese herausgestellt und eine mathematische Modellierung vorgestellt.

3.1.3. Das Repositionierungsproblem

Der wichtigste Unterschied zum DSIRP ist die Nachfragebeschaffenheit in einem BSS. Es gibt zwei verschiedene Arten von Nachfragen. Auf der einen Seite gibt es die Nachfrage nach Fahrrädern, auf der anderen die nach Fahrradstellplätzen. Dies führt dazu, dass der Füllstand in einer Station nach oben *und* unten reguliert werden muss. Beide Kapazitätsgrenzen können zu Kundenumwegen führen, wenn

ein Kunde mit Rad an eine volle Station oder ein Kunde mit Entleihwunsch an eine leere Station kommt. Ein Transportfahrzeug führt demnach auch zwei verschiedene Operationen durch. An Stationen mit Radnachfrage wird der Füllstand aufgefüllt, während er an Stationen mit Stellplatznachfrage verringert wird. Die Aufgabe des Depots verändert sich in der Hinsicht, dass es keine Lager- bzw. Produktionsstätte mehr ist. Es beherbergt lediglich die Transportfahrzeuge, die es sowohl ohne Ladung verlassen, als auch ohne Ladung zurückkehren. Diese Aspekte sind aus Pickup-and-Delivery-Problemen (PDP) bekannt. In der Literatur finden sich sowohl Forschungsbeiträge zum one-commodity Pickup-and-Delivery Travelling Salesman Problem (1-PDTSP) als auch zum one-commodity Pickup-and-Delivery VRP (1-PDVRP). Hernández-Pérez et al. [38] beschreiben das 1-PDTSP als TSP, in dem ein Transporter mit gewisser Kapazität verschiedene Städte entweder mit einem Gut beliefern oder dieses Gut dort abholen muss. Sie schlagen zwei heuristische Ansätze vor, mit denen sie Instanzen von bis zu 100 Städten in unter 15 Minuten lösen können. Hosny und Mumford [40] nutzen einen hybriden Algorithmus aus VNS und Simulated Annealing (SA) und lösen damit Instanzen mit bis zu 500 Kunden. Sie vergleichen ihre Ergebnisse mit denen von Zhao et al. [83]. Die Ergebnisse deren genetischen Algorithmus' verbessern sie, aufsteigend mit der Größe der Instanz, in 50% bis 100% der Testfälle. Martinovic et al. [47] stellen zwar das 1-PDVRP vor, beschränken dann jedoch die Anzahl der Transporter auf eins. Der vorgeschlagene iterative modifizierte SA-Ansatz löst Instanzen mit bis zu 70 Knoten. Einen Vergleich von verschiedenen Lösungsmethoden für das 1-PDVRP liefern Gunes et al. [36]. Zunächst stellen sie einen Mixed Integer Programming- und einen Constraint Programming-Ansatz vor, die das Problem allerdings nur mit bis zu 18 Kunden und einem Transporter optimal lösen können. Der Constraint-Based Local Search schafft ohne Optimalitätsgarantie Instanzen mit bis zu 130 Kunden und neun Fahrzeugen.

All diese Ansätze sind deterministischer Natur und beinhalten keine Entscheidung über die Belieferungs- oder Abholmenge. Alle Nachfragen werden als gegeben angenommen. Folglich ist keine der existierenden Modellierungen anwendbar. Die passende Modellierung des Repositionierungsproblems wäre eine stochastische und dynamische one commodity IRP-Variante mit Pickups und Deliveries (1-DSIRPPD). Für die mathematische Formulierung nutzen wir folgende Notation. Wie das IRP sei das Repositionierungsproblem auf dem Graphen $\mathcal{G}(\mathcal{V}, \mathcal{E})$ definiert. Depot und Stationen $i = 0, ..., n$ sind dabei die Knoten des Graphen. Die Lauf- bzw. Fahrtzeiten per Rad oder Transporter c_{ij}^{f}, c_{ij}^{b} und c_{ij}^{tr} sind mit den Kanten assoziiert. Jede

Station i hat eine Kapazität C_i und zu jeder Periode t einen Füllstand $b_{it} \in [0, C_i]$ von Fahrrädern. Die Transportermenge sei durch die Menge aller $v = 1, ..., V$ gegeben und die Transporter haben eine Ladekapazität von Q_v. Die Lademenge zur Periode t sei $q_{vt} \in [0, Q_v]$. Die Liefer- oder Abtransportmenge von Rädern durch Transporter v zu oder von Station i zu Periode t sei $rep_{ivt} \in \mathbb{Z}$. Dabei stehen positive Werte für Lieferungen und negative für Abholungen. Für die Lieferung oder Abholung von Rädern fällt jeweils eine fixe Servicezeit c^s pro Mengeneinheit an. Die Anzahl der Entleihungen an Station i zur Periode t ist e_{it}, die Anzahl der Rückgaben r_{it}. Dabei ist e_{it} eine poissonverteilte Zufallszahl mit Parameter λ_{it}. Die Zielortwahl nach einem Entleihereignis ist diskret auf die Stationen verteilt. r_{it} ergibt sich also aus der Summe der Entleihereignisse, die jeweils in $t - c^b_{ji}$ an allen Stationen j stattgefunden haben und i zum Ziel haben. Kann aufgrund der Kapazitätrestriktionen eine Nachfrage nicht erfüllt werden, entstehen Kundenumwege $c^{det}_{it}(b_{it})$. Abhängig davon, ob sie ein Rad oder einen Stellplatz nachfragen, laufen oder fahren sie zur nächstgelegenen Station um einen erneuten Entleih- oder Rückgabeversuch zu starten. Nicht realisierte Entleihungen oder Rückgaben werden mit ne_{it} bzw. nr_{it} bezeichnet. Wir minimieren eine kombinierte Zielfunktion aus den Transporterfahrtzeiten, den Servicezeiten und den Kundenumwegen.

$$\min \left(\sum_{i=1}^{n} \sum_{\substack{j=1 \\ i<j}}^{n} \sum_{v=1}^{V} \sum_{t=0}^{T} c^{tr}_{ij} x_{ijv} + \sum_{i=1}^{n} \sum_{v=1}^{V} \sum_{t=0}^{T} |rep_{ivt}| \cdot c^s + \sum_{i=1}^{n} \sum_{t=0}^{T} c^{det}_{it}(b_{it}) \right) \tag{3.17}$$

$$\text{u.d.N. } b_{it} = b_{it-1} - e_{it-1} + r_{it-1} + ne_{it-1} - nr_{it-1} + \sum_{v=1}^{V} rep_{ivt}, \forall i, \forall t > 0 \tag{3.18}$$

$$0 \leq b_{it} \leq C_i, \forall i, \forall t \tag{3.19}$$

$$-b_{it} \leq \sum_{v=1}^{V} rep_{ivt} \leq C_i - b_{it}, \forall i, \forall t \tag{3.20}$$

$$|rep_{ivt}| \leq \begin{cases} y_{ivt} b_{it}, \text{ für } rep_{ivt} \leq 0 \\ y_{ivt} (C_i - b_{it}), \text{ für } rep_{ivt} \geq 0 \end{cases}, \forall i \in \mathcal{V} \backslash \{0\}, \forall v, \forall t \tag{3.21}$$

$$\sum_{\substack{j=1 \\ i<j}}^{n} x_{ijvt} + \sum_{\substack{j=1 \\ i>j}}^{n} x_{jivt} = 2 y_{ivt}, \forall i, \forall v, \forall t \tag{3.22}$$

$$0 \leq \sum_{l=1}^{L} rep_{ivt_l} \leq Q_v, \forall v, L = 1, ..., |\{rep_{ivt_l}\}| \tag{3.23}$$

$$\sum_{i \in \mathcal{S}} \sum_{\substack{j \in \mathcal{S} \\ i<j}} x_{ijvt} \leq \sum_{i \in \mathcal{S}} y_{ivt} - y_{mvt}, \forall v, \forall \mathcal{S} \subseteq \mathcal{V}_v, \forall t, \forall m \in \mathcal{S} \tag{3.24}$$

$$rep_{ivt} \in \mathbb{Z}, \forall i \in \mathcal{V} \backslash \{0\}, \forall v, \forall t \tag{3.25}$$

$$x_{ijvt} \in \{0, 1\}, \forall i, j \in \mathcal{V}, \forall v, \forall t \tag{3.26}$$

$$y_{ivt} \in \{0, 1\}, \forall i \in \mathcal{V} \backslash \{0\}, \forall v, \forall t \tag{3.27}$$

Die Zielfunktion 3.17 hat eine andere Zusammensetzung als im IRP. Der operative Teil wird durch die Servicekosten erweitert. Der Strafterm ist nicht direkt auf die Bestände und nicht erfüllte Nachfragen bezogen, sondern beides wird in den Kundenumwegen zusammenfasst. Zu den Entscheidungsvariablen wird eine dritte, nämlich rep_{ivt}, hinzugefügt. Formel 3.18 entspricht den Bestandsübergängen an den Stationen. Diese sind nicht mehr nur abhängig von einer Nachfrage, sondern von zwei Nachfragen in beide Richtungen der Kapazitätsgrenzen. Hinzu kommen die Repositionierungen und die Fehlmengen von Rädern sowie Stellplätzen. 3.19 beschränkt die Bestände und 3.20 die repositionierten Mengen. Restriktion 3.21 zwingt die Repositionierungsmengen auf 0, wenn die entsprechende Station nicht besucht wird. 3.22 reguliert Zu- und Abflüsse der Transporterfahrten. 3.23 sorgt dafür, dass die Transporterkapazitäten zu keinem Zeitpunkt der Tour verletzt werden. Dafür wird der Index l so definiert, dass er die Lieferungen und Abholungen einer Tour von Fahrzeug v aufsteigend nach t sortiert. Die Subtourbedingung 3.24 wird für jede Teilknotenmenge $\mathcal{V}_v \subseteq \mathcal{V}$ einzeln benötigt. Dabei ist $\mathcal{V}_v$ die Menge der Knoten, die von Fahrzeug v angefahren werden. Die Restriktionen für die Variablen sind von 3.25 bis 3.27 festgehalten.

3.2. Komplexität

Die Vorgänge in einem Bike-Sharing-System (BSS) lassen sich sehr gut als Markovprozess (siehe Kapitel 2.3.1) darstellen. Wir definieren dafür eine Menge von Entscheidungspunkten $k = 1, ..., K$, die im Zeithorizont $t = 0, ..., T$ liegen. Dabei sind die wichtigen Variablen zum Entscheidungszeitpunkt k wie folgt belegt:

- **Zustandsvariable** - Füllstände (Fahrradbestände) b_{ik} an allen Stationen $i = 1, ..., n$, Positionen pos_v der Transportfahrzeuge $v = 1, ..., V$

- **Entscheidungsvariable** - Entscheidung d_k über Wunschfüllstände b^*_{ik}, Fahrradtransporte und deren Umsetzung im Routing
- **äußerer Einfluss** - stochastische Kundennachfrage nach Rädern oder Stellplätzen $\omega_k = \left(\sum_{t=k}^{k+1} e_{it}, \sum_{t=k}^{k+1} r_{it} \right), \forall i$
- **Zustandsübergang** - Veränderung der Füllstände und Positionen von S_k nach S_{k+1} unter Berücksichtigung der Entscheidung d_k und des äußeren Einflusses ω_k
- **Kosten- bzw. Ertragsfunktion** - Transportaufwand $c^{tr}_k(d_k) + c^s_k(d_k)$ abhängig von der Entscheidung d_k und Kundenumwege $c^{det}_k(S_k, d_k, \omega_k)$ abhängig von Zustand S_k, Entscheidung d_k und stochastischer Kundennachfrage ω_k
- **Zielfunktion** - $\min_{\pi \in \Pi} \mathbb{E} \sum_{k=1}^{K} c^{tr}_k(d^\pi_k(S_k)) + c^s_k(d^\pi_k(S_k)) + c^{det}_k(S_k, d^\pi_k(S_k), \omega_k)$, wobei π die entsprechend gewählte Entscheidungspolitik aus der Menge der möglichen Politiken Π ist.

Um die Notation übersichtlich zu halten, fassen wir den Aufwand für Routing c^{tr}_k und Auf- bzw. Abladen c^s_k zusammen und schreiben im Folgenden für beides zusammen c^{tr}_k.
Wie in Kapitel 2.3.2.1 herausgestellt, ergibt sich aus der Kombination von Zustands- und Entscheidungsraum sowie des zu berechnenden Erwartungswertes eine sehr hohe Dimensionalität. Der Zustandsraum setzt sich aus den Kombinationen der Positionen sowie der möglichen Lademengen der Transportfahrzeuge und den Kombinationen möglicher Füllstände an den Stationen zusammen. Seine Größe kann also mit

$$Q_v^{|V|} \cdot n^{|V|} \cdot \prod_{i=1}^{n} C_i \tag{3.28}$$

abgeschätzt werden. Dabei ist V die Menge der Repositionierungsfahrzeuge und Q_v die jeweilige Ladekapazität. $Q_v^{|V|} \cdot n^{|V|}$ gibt also die Anzahl an möglichen Anordnungen der Fahrzeuge im System sowie die Kombinationen an Lademengen wieder. C_i sind die Kapazitäten der Stationen. Der Entscheidungsraum umfasst die möglichen Entscheidungen über die einzelnen Belieferungen oder Abholungen kombiniert mit der zugehörigen Route. Als Anzahl möglicher Entscheidungskombinationen über

alle Stationen ergibt sich

$$\prod_{i=1}^{n} (2C_i + 1) \,. \tag{3.29}$$

Die Anzahl möglicher Tourenkombinationen ist im Fall, dass jede Station nur einmal besucht wird, durch

$$n! \cdot |V|^n \tag{3.30}$$

beschränkt. Der erste Teil des Terms gibt dabei die Anzahl der möglichen Reihenfolgen der Stationen in einer Tour wieder, während der zweite Teil die Anzahl der möglichen Verteilungen aller Stationsbesuche auf die Fahrzeuge ist. Es ist ersichtlich, dass für reale Größen von n die Menge an Zuständen nicht in angemessener Rechenzeit zu erhalten ist. Noch länger würde es dauern, die Erwartungswerte für jeden zu berechnen. Im Folgenden werden ein Dekompositions- und ein Aggregationsansatz vorgeschlagen, welche den Entscheidungsraum bzw. den Zustandsraum reduzieren können.

3.3. Dekomposition

Die hier vorgeschlagene Dekomposition dient zur Reduktion des Entscheidungsraums. Durch die Trennung des Gesamtproblems in ein Inventory Problem (IP) und ein nachgelagertes Routing teilen wir die Entscheidungsvariablen in zwei Gruppen. Im IP betrachten wir lediglich die Kombinationen an Bestandsänderungen (Komplexität wie in Formel 3.29) und nicht mehr die Binärvariablen. Das eigentliche Routing erfolgt anschließend heuristisch (siehe Kapitel 5.3). Dabei liegt die schwerwiegendere Entscheidung in der Lösung des IP, welche dann durch das Routing umgesetzt wird.
Für das IP kommt ein Ansatz der ADP zum Einsatz. Wir bedienen uns der Notation der dynamischen Programmierung aus Kapitel 2.3. Die Entscheidung d_k wird in jedem Entscheidungszeitpunkt k basierend auf der resultierenden Summe des geschätzten aktuellen Transportaufwandes und der erwarteten zukünftigen Kundenumwege und Repositionierungen getroffen. Die zu minimierende Gesamtsumme

ist durch

$$\min_{(d_k)_{k=1}^K} \sum_{k=1}^{K} c_k^{tr}(d_k) + c_k^{det}(S_k, d_k, \omega_k) \tag{3.31}$$

gegeben. Der Wert $V(S_k)$ eines Zustands S_k ist dann

$$V(S_k) = \min_{d_k \in \mathcal{D}} \left(c_k^{tr}(d_k) + \mathbb{E}\left\{ c_k^{det}(S_k, d_k, \omega_k) \right\} + \mathbb{E}\left\{ \gamma V(S_{k+1}) | S_k \right\} \right). \tag{3.32}$$

Wie in Kapitel 2.3.2.2 gezeigt wurde, bringt die Einführung eines Post-Decision-Zustandes (PDS) $S_k^{d_k}$ Vorteile mit sich. Der Wert von $S_k^{d_k}$ ergibt sich aus

$$V(S_{k-1}^{d_{k-1}}) = c_k^{det}(S_{k-1}, d_{k-1}, \omega_{k-1}) + \mathbb{E}\left\{ \min_{d_k \in \mathcal{D}} \left(\hat{c}_k^{tr}(d_k) + \gamma V(S_k^{d_k}) | S_{k-1}^{d_{k-1}} \right) \right\}. \tag{3.33}$$

Dabei trennen sich die zu minimierenden Terme in zwei Betrachtungszeitpunkte auf. Die Kundenumwege entstehen durch den äußeren Einfluss ω_k, der Routingaufwand durch die Entscheidung d_k jeweils in der Periode k. Bei der Betrachtung des PDS $S_k^{d_k}$ liegt aber die Entscheidung in der Vergangenheit, während der äußere Einfluss noch bevor steht. In seinen Wert fließt also die aktuelle Entscheidung nicht mehr ein, sondern ausschließlich die zukünftigen. Die Kundenumwege aus ω_k sind aber in jedem Fall dem PDS $S_k^{d_k}$ anzurechnen. Wenn $\overline{V}\left(S_k^{d_k}\right)$ eine Abschätzung des Wertes des PDS $S_k^{d_k}$ ist und wir uns in Iteration m eines ADP-Algorithmus befinden, dann lässt sich die Entscheidung nach folgender Regel auswählen:

$$\hat{v}_k^m = c_k^{det}(S_{k-1}, d_{k-1}, \omega_{k-1}) + \min_{d_k \in \mathcal{D}} \left(\hat{c}_k^{tr}(d_k) + \gamma \overline{V}\left(S_k^{d_k}\right) \right) \tag{3.34}$$

Da zum Entscheidungszeitpunkt k die entsprechenden Umwege schon vorliegen, wirken sie sich nicht mehr auf die Entscheidung aus, sehr wohl aber auf den Wert des vorausgehenden PDS $S_{k-1}^{d_{k-1}}$. Mit der Lösung dieses deterministischen Minimierungsproblems können wir nun die Schätzwerte der PDS nach der Formel 2.14 aktualisieren:

$$\overline{V}^m(S_{k-1}^{d_{k-1}}) = (1 - \alpha_{m-1})\overline{V}^{m-1}(S_{k-1}^{d_{k-1}}) + \alpha_{m-1}\hat{v}_k^m.$$

Da d_k ausschließlich über die Bestandsregulierungen entscheidet, können der Aufwand für Routing und Service nur abgeschätzt werden: $\hat{c}_k^{tr}(d_k) \approx c_k^{tr}(d_k)$. Dies

geschieht parallel zum ADP-Verfahren. Während durch iterative Simulation und Aktualisierung der PDS-Werte die Auswahl der besten Entscheidung gelernt wird, kann der Routingaufwand, der aus Entscheidung d_k resultiert, durch multiple heuristische Realisierung in den Simulationen angenähert werden. Sei $(c_k^{tr}(d_k))^m$ der Aufwand der m-ten Realisierung der Entscheidung d_k in der Simulation. Den m-ten Schätzwert $(\hat{c}_k^{tr}(d_k))^m$ erhält man dann durch

$$\left(\hat{c}_k^{tr}(d_k)\right)^m = (1-\alpha_{m-1})\left(\hat{c}_k^{tr}(d_k)\right)^{m-1} + \alpha_{m-1}\left(c_k^{tr}(d_k)\right)^m, \tag{3.35}$$

wobei α_m wieder ein entsprechend glättender Faktor ist.
In Kapitel 4 wird diese Dekomposition des IRP in Hinblick auf die Informationsflüsse im Kontext eines betrieblichen Informationssystems eingeordnet.

3.4. Aggregation

Sowohl Zustands- als auch Entscheidungsraum des Repositionierungsproblems in realen Instanzgrößen müssen reduziert werden, damit sie für die ADP handhabbar werden. Dabei muss auf ein Gleichgewicht zwischen Genauigkeit und Berechenbarkeit geachtet werden. Liegen zu viele Zustände und Entscheidungen vor, so ist der Rechen- und Speicheraufwand hoch. Die Lernleistung des Algorithmus ist womöglich schlechter, weil der einzelne Zustand zu selten erreicht wird. Sind die Intervalle zu grob gewählt, dann verlieren sie an Aussagekraft. Zwar werden die Zustände häufig erreicht, aber möglicherweise mit unterschiedlichen Ausgängen, da sehr unterschiedliche Füllstände in eine Kategorie fallen könnten. Um eine ausreichende Approximation in angemessener Zeit zu erhalten, reduzieren wir Zustands- und Entscheidungsraum auf zwei verschiedene Arten. Zum einen werden die Zustände $S_{ik}^{d_{ik}}$ und Entscheidungen d_{ik} jeder Station einzeln betrachtet. Zum anderen werden die Füllstände sowie die Füllstandsänderungen zu grobkörnigen Intervallen aggregiert. Die Einzelbetrachtung der Stationen führt zwar dazu, dass wir für jede Station ein IP zu lösen haben, verringert aber den kombinatorischen Umfang drastisch auf

$$n \cdot \max_{i=1}^{n} \{\mathrm{ivl}_i\}. \tag{3.36}$$

Mit ivl_i wird die Anzahl der betrachteten Intervalle an Station i bezeichnet. Die Größe des Zustandsraumes lässt sich folglich für jedes der einzelnen IP mit der Menge der Füllstandsintervalle beschreiben. Die Position der Transportfahrzeuge

wird für die einzelne Station nicht mehr benötigt. Es empfiehlt sich, den Entscheidungsraum ebenso grobkörnig zu aggregieren wie den Zustandsraum, damit die Füllstandsänderungen zu den entsprechenden Intervallen an der Station passen. Außerdem hätte eine Entscheidung auf ein Fahrrad genau gegenüber der Grobkörnigkeit des Zustandsraumes wenig Sinn, da kleine Veränderungen keine Zustandsänderungen hervorrufen würden. Hinzu kommt noch, dass sie in vielen Fällen zur Unlösbarkeit führen könnte. Die Größe des Entscheidungsraumes ist also durch die Menge an Veränderungsoptionen pro Station gegeben:

$$2 \cdot \mathrm{ivl}_i + 1. \tag{3.37}$$

Kapitel 5 erläutert ausführlich, wie und an welcher Stelle im Modell unseres betrieblichen Informationssystems die Dekomposition sowie die Aggregation stattfinden.

4. Einbettung in die Modellierung betrieblicher Informationssysteme

Nachdem im vorhergehenden Kapitel 3 das Repositionierungsproblem mathematisch formuliert wurde, betrachten wir nun unseren Planungsansatz zur Problemlösung als Teil eines Informationssystems (IS) im betrieblichen Sinne. Ferstl und Sinz [31] vergleichen das betriebliche IS mit dem menschlichen Nervensystem, welches alle Aktivitäten lenkt. Die Aufgaben dieses IS sind Planung, Steuerung und Kontrolle der betrieblichen Prozesse sowie der Interaktionen mit der Umwelt. Auf eine ähnliche Art beschreibt Schneeweiß [70] verteilte Entscheidungssysteme (DDMS, Distributed Decision Making Systems) im betrieblichen Umfeld. Diese bestehen aus unterschiedlichen Hierarchieebenen, welche sich gegenseitig beeinflussen können. Bezogen auf ein BSS übernimmt die Repositionierung die Aufgabe der Kontrolle und Steuerung des Systemablaufs. Das Repositionierungsproblem lässt sich folglich gut in den Kontext der Modellierung betrieblicher IS einbetten. Kapitel 4.1 führt von den beiden zugrunde liegenden Modellen zu einer verfeinerten, auf unser Problem angepassten, Darstellung eines Lenkungsebenenmodells. Die einzelnen Ebenen werden allgemein mit Bezug auf den Betrieb eines BSS beschrieben. Diese allgemeinen Beschreibungen bilden den Übergang zu Kapitel 5. Dort erfolgt die praktische Umsetzung des Ansatzes.

4.1. Lenkungsebenenmodell

In Anlehnung an das Lenkungsebenenmodell von Ferstl und Sinz [31] kann die Dekomposition des vorliegenden Repositionierungsproblems in hierarchisch geordnete Ebenen gesplittet werden. Abbildung 4.1 verdeutlicht die zugrunde liegenden Ebenen und deren Zusammenhänge. Dabei dient das operative Informationssystem der unmittelbaren Lenkung eines Basissystems, während das strategische In-

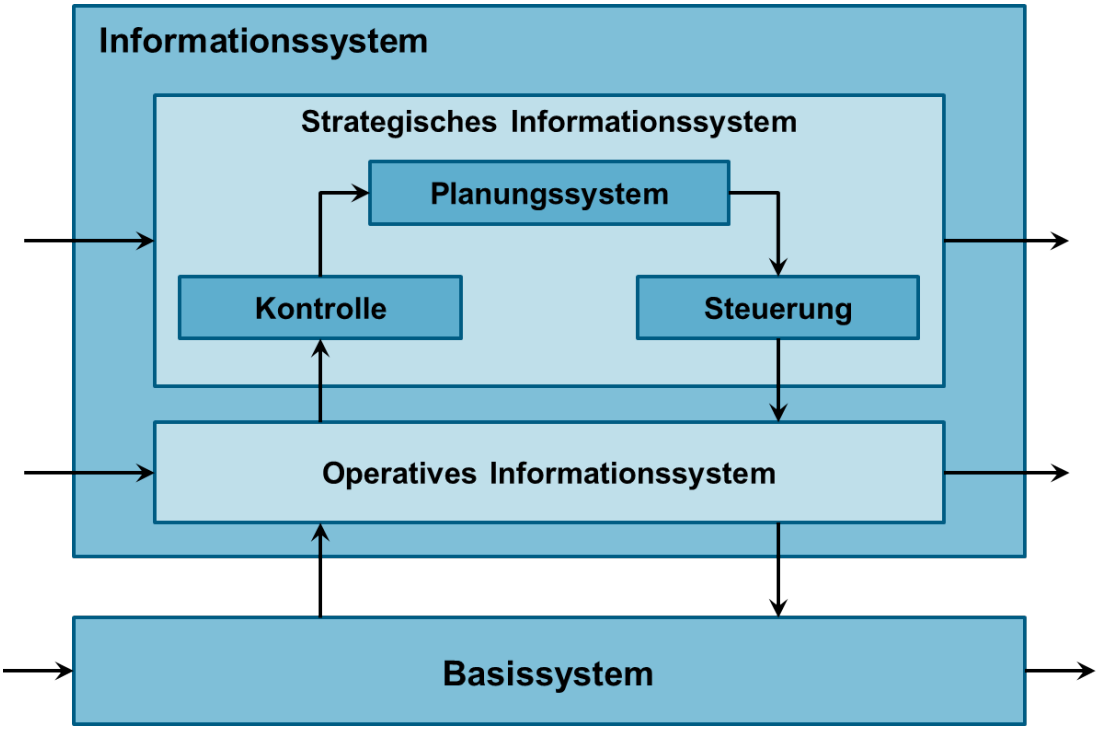

Abbildung 4.1.: Lenkungsebenenmodell nach Ferstl/Sinz [31]

formationssystem die Lenkung des vollständigen Systems auf grobkörniger, also aggregierter Ebene zur Aufgabe hat. Hier spiegeln sich die Hierarchieebenen eines Unternehmens wider. Beispielsweise wären in einem Produktionsbetrieb die Mitarbeiter auf operativer Ebene für jedes einzelne Produkt verantwortlich und könnten direkt und genau eingreifen. Das strategische Management benötigt zur Führung des Unternehmens jedoch grobkörnige Informationen. Managemententscheidungen finden folglich auf derselben grobkörnigen Ebene statt. Innerhalb des strategischen Informationssystems werden Planung, Steuerung und Kontrolle unterschieden. Die Aufgabe des Planungssystem ist es, auf Grundlage von Informationen aus den unteren Ebenen und von außen strategische Entscheidungen zu treffen. Diese dienen wiederum als Führungsgrößen für die Entscheidungen auf den anderen Ebenen. Die benötigten Informationen werden von der Kontrolle aus einem Basissystem gewonnen und entsprechend der Anforderungen des Planungssystems vorverarbeitet. Die Steuerungsaufgabe besteht in der Umsetzung der Entscheidungen, die das Planungssystem vorgibt.

Einen anderen Fokus setzt Schneeweiß bei seiner Darstellung eines zweistufigen DDMS, wie Abbildung 4.2 zeigt. Hier wird verdeutlicht, welche Beziehungen zwischen den Ebenen herrschen. Die hierarchisch höchste Position hat das Top-Level. Dort wird die erste Entscheidung getroffen. Auf die resultierenden Anweisungen

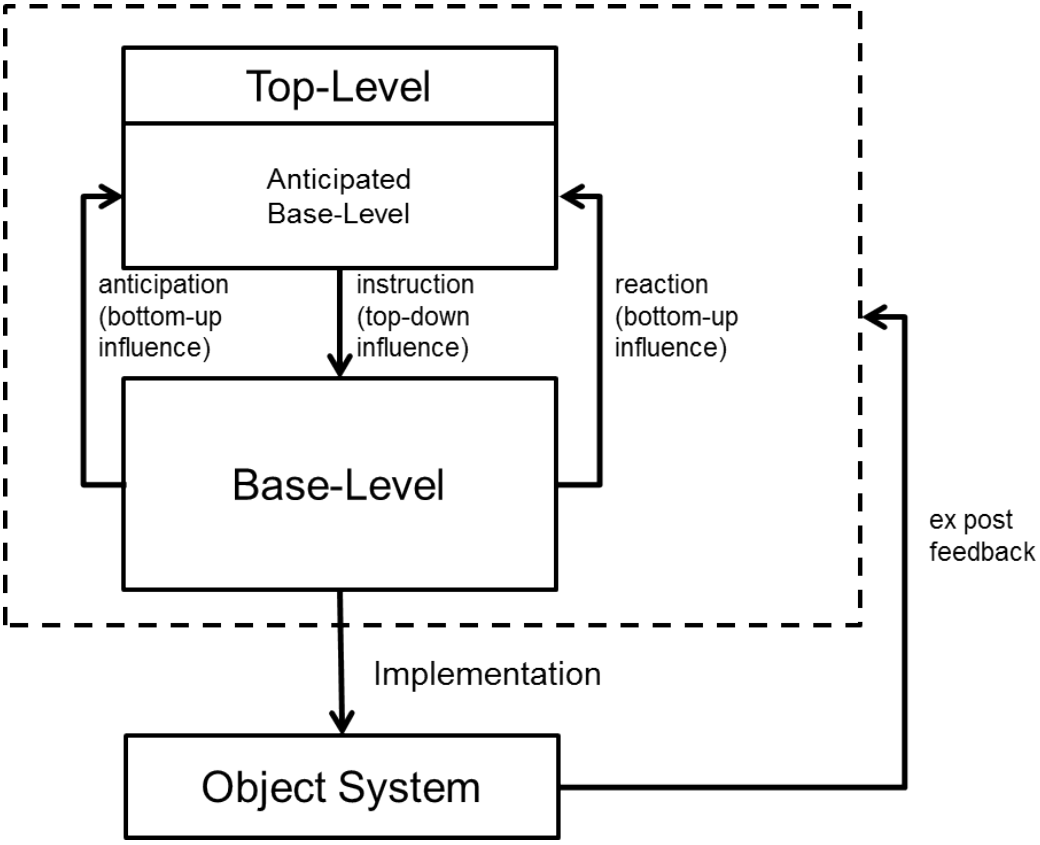

Abbildung 4.2.: Zweistufiges DDMS nach Schneeweiß [70]

aufbauend kann das Base-Level entscheiden und die Reaktion sowie Informationen zur Antizipation an das Top-Level zurück melden. Die Implementierung erfolgt in der dritten Ebene, welche bei Schneeweiß mit Object System bezeichnet wird. Dieses kann durch ein ex-post-Feedback die Entscheidungen der oberen Ebene beeinflussen.

Für unser angepasstes Lenkungsebenenmodell (siehe Abb. 4.3) nutzen wir Teile von beiden vorangehenden Abbildungen. Die Planungsebene nimmt die Position des Top-Levels ein, während die Aufgaben der Steuerung und der Kontrolle hierarchisch zwischen dem Basissystem und der Planung stehen. Sie bekommen in unserem speziellen Lenkungsebenenmodell für das Repositionierungsproblem eine eigene Ebene mit dem Namen *Steuerung* und nehmen die Position des Base-Levels ein. So werden die langfristigen Entscheidungen auf aggregierter Ebene von den kurzfristigen feinkörnigen sauber getrennt. Die Ebene des Betriebs entspricht dem Basissystem bzw. dem Object System. Eine wichtige Ergänzung sind die Informationsflüsse zwischen den Ebenen, welche jeweils Einfluss auf die benachbarte Ebene nehmen. Es ergibt sich ein Informationskreislauf, welcher einen sinkenden

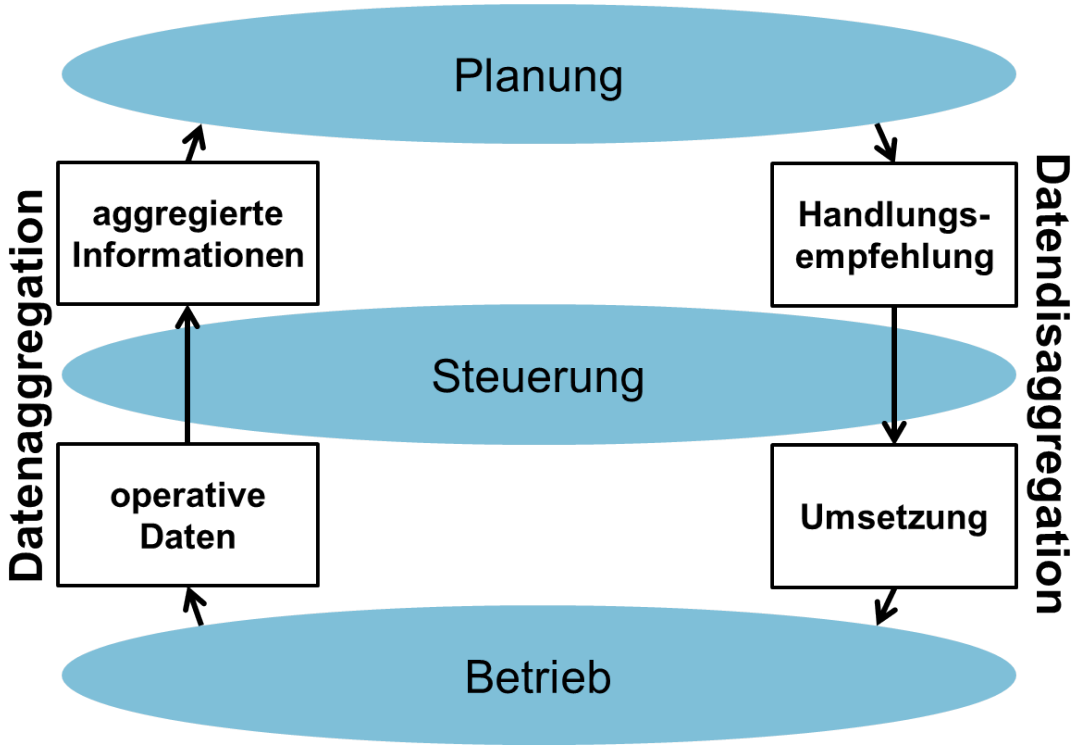

Abbildung 4.3.: Verfeinertes Lenkungsebenenmodell

Detailgrad auf der einen Seite und einen wachsenden auf der anderen hat. Deutlich ist die hierarchische Datenverdichtung erkennbar, wie sie in Planungssystemen auftritt. Mertens et al. [50] stellen sie in abstrahierter Form ähnlich der Abbildung 4.4 dar.

In den folgenden Unterkapiteln werden die einzelnen Bestandteile des verfeinerten

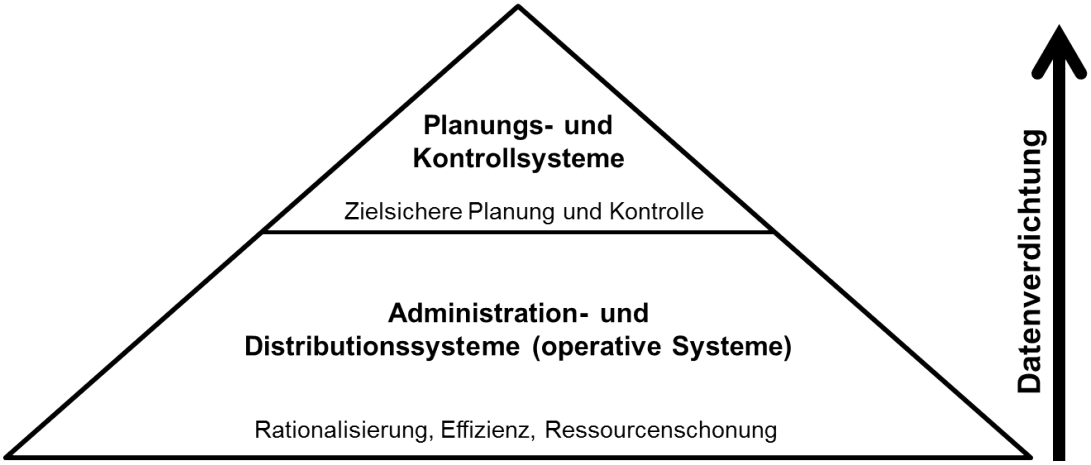

Abbildung 4.4.: Datenverdichtung in Planungssystemen in Anlehnung an Mertens [50]

Modells erläutert.

4.1.1. Betrieb

Das Basissystem bildet der Betrieb des Bike-Sharing-Systems. Dort spielen sich alle Bewegungen im Systembetrieb ab. Es liefert operative Daten wie Stationsfüllstände oder Kundenfahrten. Repositionierungsfahrten oder andere betreiberseitige Bewegungen, die durch die Planung ausgelöst werden, finden sich hier ebenso wieder.

4.1.2. Steuerung

Das Steuerungslevel umfasst die Aufgaben der Kontrolle und der Steuerung. Wie in Abbildung 4.3 verdeutlicht, finden auf der Datenaggregationsseite und der Datendisaggregationsseite gegenläufige Informationstransformationen statt. Die Aggregationsseite wandelt aus dem Betrieb gewonnenen Daten in Informationen um, die für die Planung notwendig sind. Die Disaggregationsseite hat die umgekehrte Aufgabe, die Informationen aus dem Planungslevel in der Betriebsebene, also dem Basissystem, zu implementieren. Auf beiden Seiten finden Entscheidungen statt, die der Planung vorgegriffen bzw. nachgestellt werden. Durch ersteres wird die Planung beeinflusst, während letzteres das Planungsergebnis interpretiert. So wird auf der Aggregationsseite die Grob- oder Feinkörnigkeit der Füllstandsintervalle festgelegt, während der Dissaggregationsseite bei der Umsetzung der Planungsergebnisse Entscheidungsspielräume bezüglich der Disaggregation und der Implementierung bleiben.

4.1.3. Planung

Das Planungslevel bietet die Möglichkeit, durch Optimierung auf aggregierten Systeminformationen kontrolliert in den Systembetrieb einzugreifen. Aus auf dem Steuerungslevel vorverarbeiteten Informationen kann es eine Handlungsempfehlung ableiten, welche dann durch die Steuerung auf den unteren Ebenen umgesetzt werden kann.

Im folgenden Kapitel 5 findet das hier modellierte betriebliche IS Anwendung. Die einzelnen Ebenen des Modells werden in eine Methodensicht überführt. Die Implementierung erfolgt in Java.

5. Umsetzung und Implementierung der Dekomposition

Dieses Kapitel überführt die Modellierung aus dem vorhergehenden Kapitel 4 in eine implementierbare Darstellung. Die einzelnen Unterkapitel beschreiben die praktischen Umsetzungen der Optimierungsvorgänge und deren Implementierung in Java.
Passt man das Lenkungsebenenmodell aus Abbildung 4.3 an das Repositionierungsproblem an, entsteht folgende Methodensicht auf das Modell (siehe Abb. 5.1).

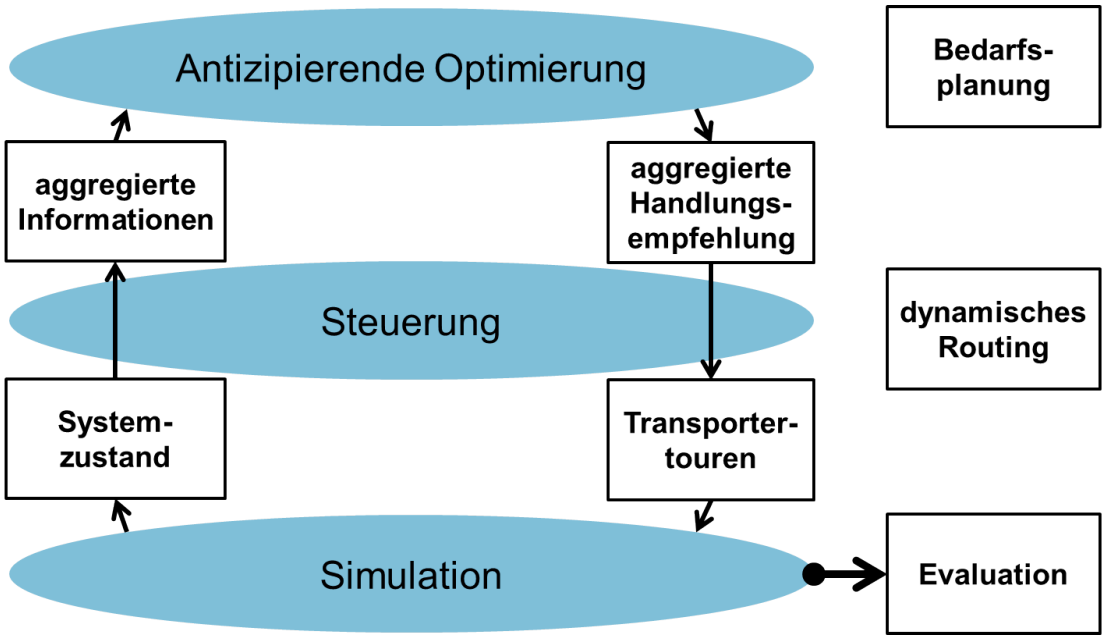

Abbildung 5.1.: Methodensicht

Auf die Inhalte, Aufgaben und Implementierung der einzelnen Ebenen wird nachfolgend ausführlich eingegangen.

5.1. Simulation

Der Betrieb des BSS wird im vorliegenden Fall von einer *eventbasierten* Simulation abgebildet. Eventbasiert bedeutet, dass die Simulation mit dem Eintreten von Ereignissen statt in festen Zeitschritten durchlaufen wird. Veränderungen werden durch verschieden definierte Events hervorgerufen. Folglich genügt es, die Simulation bei jedem Eintritt eines Events zu betrachten. Mittels eines Java-Programms werden die Stationen $i \in I$ des BSS modelliert. Jedes Objekt der Klasse `Station` enthält unter anderem beschreibende Attribute für den Standort, die Kapazität C_i und den Bestand b_{it} in Zeitpunkt t. Das Herzstück der Simulation bildet die Eventqueue `events` mit dem zugehörigen Interface `Event`. In Abbildung 5.2 sind die Diagramme von `Event` und einer beispielhaften Implementierung `RandomEvent` dargestellt. Die Eventqueue enthält nach Eintrittszeitpunkt `time` geordnete Events. Ist sie nicht leer, so wird immer das erste Element der Queue herausgenommen (`poll()`) und dessen Funktion `run()` ausgeführt. Mit den verschiedenen Implementierungen des Interfaces `Event` können kundenseitige Entleih- oder Rückgabebewegungen sowie betreiberseitige Repositionierungsbewegungen umgesetzt oder auch das Simulationsende herbeigeführt werden. Unverzichtbar für eine realitätsnahe Simulation eines BSS ist das stochastische Auftreten der Kunden. Hierfür werden entsprechende Informationen über die Entleih- und Zielortwahrscheinlichkeiten für die einzelnen Stationen benötigt. In dieser Simulation werden diese Informationen aus historischen Daten des BSS Citybike in Wien [35] ermittelt. Da die Wahrscheinlichkeiten über den Tag variieren [80], wurde für jede Station i pro Stunde t eines durchschnittlichen Sommerwerktages ein Exponentialverteilungsparameter

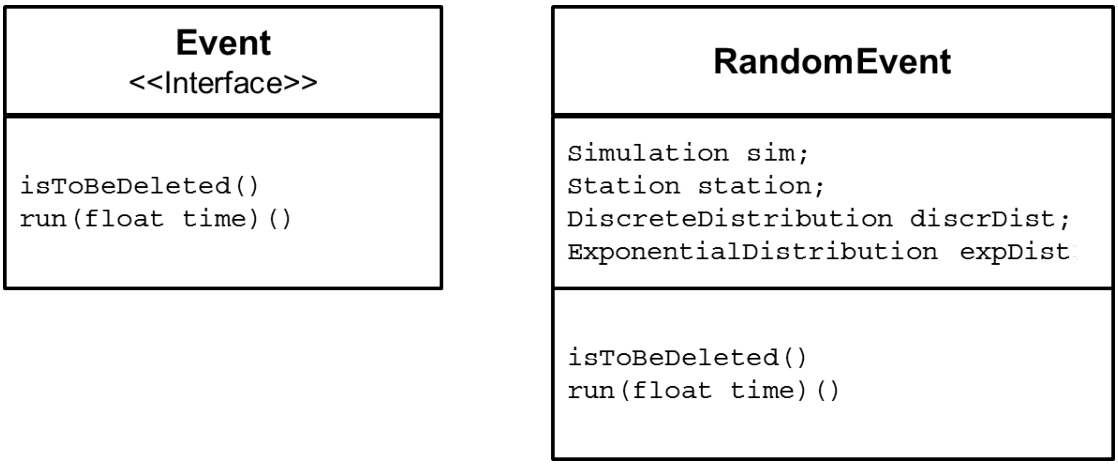

Abbildung 5.2.: Interface Event und Klasse RandomEvent

λ_{it} für die Ereigniseintrittsrate ermittelt. Von einem Zeitpunkt t_0 aus gesehen, berechnet sich der Zeitpunkt der nächsten Entleihung durch

$$t_{\text{next}} = t_0 + \left(-\frac{1}{\lambda_{it_0}} \log\left(1 - z\right) \right) \tag{5.1}$$

mit einer generierten Zufallszahl $z \in [0, 1]$, falls t_{next} in dasselbe Zeitintervall wie t_0 fällt (formal: $\lfloor t_{\text{next}} \rfloor = \lfloor t_0 \rfloor$). Eine Herleitung hierzu findet sich in Anhang A.1. Liegt der nächste Entleihzeitpunkt außerhalb der aktuellen Stunde, muss berücksichtigt werden, dass die Eintrittswahrscheinlichkeiten für verschiedene Stunden unterschiedlich sind. Darum wird iterativ für $n = 1, 2, \ldots$

$$t_n = \lfloor t_0 \rfloor + n \tag{5.2}$$

und

$$t_{\text{next}} = t_n + \left(-\frac{1}{\lambda_{it_n}} \log\left(1 - z\right) \right) \tag{5.3}$$

solange berechnet bis $\lfloor t_{\text{next}} \rfloor = \lfloor t_n \rfloor$. Wird diese Bedingung erfüllt, ist t_{next} der Zeitpunkt des nächsten Entleihevents. Dieses wird dann mit der Funktion `enqueue(Event)` in die Eventqueue einsortiert, und zum Eintrittszeitpunkt ausgeführt.
Für die Verteilung der Zielortwahl wurde für jede Stunde t und Station i eine diskrete Verteilung angenommen. Dabei sind die Wahrscheinlichkeiten proportional zum Auftreten bestimmter Flüsse zwischen den Stationen verteilt und auf nicht gefahrene Strecken entfällt eine kleine Restwahrscheinlichkeit. Zu jeder Kombination aus i und t ist ein Array mit kumulierten Zielwahrscheinlichkeiten hinterlegt. Mit einer Zufallszahl $z \in [0, 1]$ wird der Rückgabeort zur entsprechenden Entleihung bestimmt. Die Zeiten, die ein Kunde braucht, um von der einen zur anderen Station zu gelangen, basieren im Falle des realen Systems auf Fahrtzeiten von Google Maps. Für die künstlich erzeugten Systeme liegen Luftliniendistanzen zugrunde. Für Fahrtzeiten mittels Auto oder Rad sowie für Laufzeiten, die der Kunde zu Fuß zurücklegt, ist je eine Tabelle hinterlegt. Mithilfe dieser Tabellen kann der Zeitpunkt des Erreichens der Zielstation ermittelt werden, an dem zum Beispiel in der Folge einer Entleihung ein Rückgabeevent stattfindet, welches dann wieder der Eventqueue hinzugefügt wird.
Um die Zulässigkeit der Bewegungen zu gewährleisten, müssen Ausnahmen implementiert werden. Falls die Kapazitätsgrenzen einer Station erreicht sind, können Entleih- bzw. Rückgabeevents nicht stattfinden. Damit die entsprechende Nach-

frage nicht verloren geht, wird hier angenommen, dass die Kunden zur nächstgelegenen Station laufen oder fahren, um dort einen erneuten Entleih- oder Rückgabeversuch zu unternehmen. Je nachdem, welche Werte für den Nutzer von Bedeutung sind, können während der Simulation mit dem `Logger` Aktionen oder Zeiten aufgezeichnet werden. Auf deren Basis können die Simulationsdurchläufe ausgewertet werden. Das Ende der Simulation wird durch den Zeitparameter `simulatedDays` festgelegt. Zu Beginn der Simulation wird ein `EndEvent` in die Eventqueue zum gewünschten Endzeitpunkt eingefügt. Ab diesem Zeitpunkt werden keine neuen Aktionen mehr betrachtet, während die angefangenen Aktionen aber noch vollendet werden, damit die zugehörigen Auswirkungen, zum Beispiel Kundenumwege, nicht verloren gehen.
Mit diesen Elementen ist die Simulation nun lauffähig und kann ein BSS mit stochastischen Kundenbewegungen modellieren. Transportfahrzeuge oder andere betreiberseitige Einflüsse sowie deren Planung können modular hinzugefügt werden. Es wird für jeden Einfluss eine entsprechende Implementierung von `Event` definiert. Die Modularisierung des Java-Programms lässt es zu, dass das Steuerungs- und das Planungslevel ohne Eingriff in den Simulationsmechanismus integriert werden können. Wie genau dies in dieser Simulation realisiert wurde, wird in den folgenden Unterkapiteln gezeigt.

5.2. Datenaufbereitung

Das Steuerungslevel wurde im vorliegenden Fall auf zwei verschiedene Arten verwendet. Die erste ist zur Generierung myopischer Repositionierungstouren, die Daten der Simulation und vorgegebenen Regeln benötigen, die andere zur Datenaufbereitung für die Planung sowie der Umsetzung deren Handlungsempfehlung. Schauen wir uns zunächst die Datenversorgung an:
Die Datenaufbereitung beschränkt sich im Fall der myopischen Touren auf eine Extraktion der Stationsbestände aus der Simulation. Weiterhin werden eine oder mehrere Regeln benötigt, nach der die Entscheidung über die Transporte der Räder getroffen werden soll. Eine mögliche Regel kann zum Beispiel sein, leere Stationen mit bestimmten Mengen zu befüllen und volle Stationen von bestimmten Mengen an Fahrrädern zu befreien. So erhält man leicht einen Bedarfsvektor. Dieser wird, wie in Kapitel 5.3, mittels eines Tourenplans für die Transporter umgesetzt.
Dient die Steuerung als Vor- und Nachbereitung für das Planungslevel, so sind schon auf dieser Stufe Entscheidungen unabhängig von der Optimierung selbst zu

treffen. Die Datenaufbereitungsseite benötigt eine Vorgabe bezüglich der Aggregationsstufe, während die Art und Weise der Disaggregation zur Einbindung in die Simulation zu der Interpretationsaufgabe gehört. Beide Entscheidungen beeinflussen die Güte der Optimierung, weswegen sie mit der Optimierungsmethodik abzustimmen sind. Die Komplexität der Approximativen Dynamischen Programmierung, welche hier zur Optimierung eingesetzt wird, übersteigt mit zunehmender Größe der Zustandsräume schnell realistische Rechenzeit- und Speicherkapazitäten (siehe Kapitel 2.3). Zudem führt, wie in Kapitel 3.4 beschrieben, eine zu feine Betrachtung möglicherweise zu vielen ähnlichen Zuständen. Wird aber zu grob aggregiert, so kann das Ergebnis der Optimierung sehr ungenau und schwierig zu interpretieren werden. Es muss folglich ein Mittelmaß gefunden werden. Der Simulation wurde eine Konverter-Funktion `convertState` hinzugefügt, welche gesteuert durch den Parameter `numStates` die Bestände an den Stationen in eine der `numStates` Intervalle ivl übersetzt, mit $ivl = 0, ..., \texttt{numStates} - 1$. Dabei sind die Intervalle jeweils annähernd gleichgroß (gerundet auf ganze Zahlen). Die Zuordnung eines Bestands b_i zu einem Intervall erfolgt nach

$$ivl_i = \left\lfloor \frac{b_i \cdot \texttt{numStates}}{C_i - 1} \right\rfloor . \tag{5.4}$$

Damit erhalten wir aus den auf einzelne Fahrräder genauen Bestandsangaben einen groben Vektor von Bestandsintervallen für jede Station i. So reduziert sich der Zustandsraum für jede Station auf `numStates` pro Stunde. Der Entscheidungsraum wird durch einen zweiten Parameter `numDecisions` festgelegt. Er drückt aus, um wieviele Intervalle der Zustand einer Station verändert werden soll. Eine ungerade natürliche Zahl ist hier angebracht, um sowohl gleiche Veränderungen nach oben und unten zuzulassen als auch die Option für keine Veränderung zu haben: $d \in \left\{ -\lfloor \frac{numDecisions}{2} \rfloor, ..., 0, ..., \lfloor \frac{numDecisions}{2} \rfloor \right\}$. Mit diesen zwei Parametern wird also vorgegriffen, auf welcher Aggregationsebene das Planungslevel agiert und auch die entsprechende Lösung liefert. Bei der Disaggregation zur Einbindung der Lösung in die Simulation muss nun aus der groben Handlungsanweisung auf eine Fahrrad-genaue Transportmenge geschlossen werden. Diese wird hier durch die Intervallspannweite $\frac{C_i}{\texttt{numStates}}$ der jeweiligen Station i bestimmt. So erhält man einen Bedarfsvektor, auf dessen Grundlage analog zum myopischen Ansatz die Touren der Transporter gebildet werden.

5.3. Umsetzung der Touren

Die Umsetzung der Bedarfe in Touren erfolgt in zwei Schritten. Zunächst wird der Bedarfsvektor ausgeglichen (`findBestFeasible()`). Dies ist notwendig, damit die Nachfrage gleich der Angebotsmenge an Fahrrädern ist. Andernfalls müssten während der Touren Räder generiert werden oder verschwinden. Je nachdem, ob wir myopisch oder antizipierend planen, können wir für den Ausgleich verschiedene Informationen nutzen. Der myopische Ansatz vergleicht die relativen Füllstände an den Stationen. Diese dienen zur Bewertung der Dringlichkeit. Je dichter die Füllstände an den Stationskapazitätsgrenzen sind, desto dringender ist eine Aktion. Die am wenigsten dringenden Aktionen können dann vermindert oder sogar gestrichen werden. Im antizipierenden Fall (siehe Kapitel 5.4) kann auf die Planungsinformationen zurückgegriffen werden. Jedem Zustand ist ein bestimmter Wert zugeordnet. Dadurch können wir auch die Wertverluste durch nicht getätigte Aktionen bestimmen. Folglich können die Repositionierungsaktionen, die den geringsten Wertezuwachs bringen, vermindert oder gestrichen werden.
Nach dem Ausgleich werden die Bedarfe mit einem angepassten Greedy-Algorithmus in einer Tour geordnet (`getInitialRoute(demands)`). Gewählt wird immer die am schnellsten erreichbare Nachfrage. Dabei wird darauf geachtet, dass die Ladekapazität der Fahrzeuge nicht überschritten wird. Die Tour wird dann, falls sie nicht den Restriktionen bezüglich der Länge entspricht, in mehrere kürzere Touren gesplittet, um die Zeitrestriktionen einzuhalten (`splitRoutes(initialRoute)`). Sollte eine Teilung nicht möglich sein, so wird eine längere Tour in Kauf genommen. Dabei muss beachtet werden, dass der entsprechende Transporter für die nächste Periode evtl. nicht zur Verfügung steht. Auf die so generierte Tourenliste werden nacheinander verschiedene angepasste Verbesserungsheuristiken angewendet. Es handelt sich um Tauschoperatoren, die einzelne Nachfragen innerhalb der Touren (`doAllInnerChanges(route)`) bzw. zwischen den Touren tauschen (`optSwapBetweenRoutes(routes)`). Ein weiterer Operator verschiebt Nachfragepaare von einer Tour in eine andere (`optSwapPairsBetweenRoutes(routes)`). Für jeden der Tauschoperatoren legt ein Eingabeparameter `numOperatorname` die Anzahl der Durchläufe fest. Die Abläufe der Verbesserungsverfahren sind im Algorithmenkapitel B.1 im Anhang zu finden. Jede Tour der ermittelten Tourenliste löst die Fahrt eines Transporters aus, sofern einer verfügbar ist. Dies geschieht mittels eines `TransportEvent`, dem die entsprechende Tour sowie die Ankunftszeit an der ersten Station übergeben werden. Das Event wird in die Eventqueue

eingefügt, wo es dann zur entsprechenden Zeit abgearbeitet wird. Auch hier wird auf die Zulässigkeit der Aktionen geachtet. Sollte in seltenen Fällen eine Lieferung oder ein Abtransport nicht möglich sein, werden entsprechende Ausweichaktionen ausgelöst.

5.4. Antizipierende Optimierung für das Inventory Problem

Für die Lösung des IPs auf der Planungebene wird die ADP vorgeschlagen. Wie in den Algorithmen aus Kapitel 2.3 beschrieben, kann mittels Simulation der Zustandsverläufe ein Sample möglicher Entscheidungsauswirkungen erzeugt werden. Kapitel 5.1 erläutert, wie die Auswirkungen aufgezeichnet und zur Bewertung der Zustände und der Entscheidungen herangezogen werden können.
Zunächst erfolgt die Einordnung der vorliegenden Größen in den Markov-Prozess. Der Zeithorizont wird in gleichlange Perioden unterteilt, zu deren Anfang jeweils ein Entscheidungszeitpunkt k steht. Durch die Dekomposition des Gesamtproblems (siehe Kapitel 3.3) sowie die Aggregation auf dem Steuerungslevel (siehe Kapitel 3.4) sind die Zustands- und Entscheidungsräume festgelegt. Der Zustand S_{ik} ebenso wie der Post-Decision-Zustand $S_{ik}^{d_{ik}}$ der Station i zum jeweiligen Entscheidungszeitpunkt k liegt im Wertebereich von ivl, also $S_{ik} \in \{0, ..., \texttt{numStates} - 1\}$ (siehe Kapitel 5.2). Für den Entscheidungsraum $\mathcal{D}$ gilt $|\mathcal{D}| = \texttt{numDecisions}$. Folglich beschreiben S_{ik} und $S_{ik}^{d_{ik}}$ den aggregierten Füllstand an Station i vor bzw. nach der Entscheidung d_{ik}.
Zur Entscheidungsunterstützung ist im Simulationsprogramm eine Tabelle mit Werten $\overline{V}\left(S_{ik}^{d_{ik}}\right)$ für alle $S_{ik}^{d_{ik}}$ hinterlegt. Die Menge an Informationen auf denen die Werteapproximationen basieren, wächst mit der Anzahl an Trajektorien. Die Entscheidungen und deren Auswirkungen werden simulativ realisiert und ausgewertet. Das jeweilige Wissen über die Auswirkungen der Entscheidungen in vorherigen Trajektorien fließt, durch ein Update der Werte (siehe Formel 2.14), in den folgenden Trajektorien mit in die Entscheidungsfindung ein. Für das Repositionierungsproblem wird mit Algorithmus 3 der Algorithmus 2 aus Kapitel 2.3.2.2 angepasst.
Mit der jeweils in m aktuellen Wertetabelle aller $\overline{V}^m(S_{ik}^{d_{ik}})$ sowie den aktuellen Schätzwerten des Repositionierungsaufwandes $\hat{c}_{ik}^{tr}$ kann in jeder Trajektorie für jeden Systemzustand S_{ik} eine entsprechend empfohlene Entscheidung d_{ik} ge-

Algorithmus 3: ADP-Algorithmus für das IP im BSS

Initialisierung:
Initialisiere $V^0(S_{ik}^{d_{ik}})$, $\forall S_{ik}^{d_{ik}}$.
Setze M.
$m = 1$
Initialisiere S_{i0}.
while $m \leq M$ **do**
  Generiere Zufallsnachfragen ω^m.
  for $k = 1, ..., K$ **do**
    for $i = 1, ..., n$ **do**
      Löse

$$\hat{v}_{ik}^m = c_{ik}^{det}(S_{ik-1}, d_{ik-1}, \omega_{ik-1}) + \min_{d_{ik} \in \mathcal{D}_i} \left(\hat{c}_{ik}^{tr}(d_{ik}) + \gamma \overline{V} \left(S_{ik}^{d_{ik}} \right) \right) \tag{5.5}$$

      mit $d_{ik}^{m*} = \underset{d_{ik} \in \mathcal{D}_i}{\operatorname{argmin}} \left(\hat{c}_{ik}^{tr}(d_{ik}) + \gamma \overline{V} \left(S_{ik}^{d_{ik}} \right) \right)$
      if $k > 1$ **then**
        Aktualisiere $\overline{V}^{d_{ik}m-1}(S_{ik-1}^{d_{ik}})$:
        $\overline{V}^{d_{ik}m}(S_{ik-1}^{d_{ik}}) = (1 - \alpha_{m-1})\overline{V}^{d_{ik}m-1}(S_{ik-1}^{d_{ik}}) + \alpha_{m-1}\hat{v}_{ik}^m$
      end
    end
    Gehe zum nächsten Post-Decision-Zustand über:
    $S_{ik}^m \xrightarrow{d_{ik}^{m*}} S_{ik}^{d_{ik}m}$
    Gehe zum nächsten Zustand über:
    $S_{ik}^{d_k m} \xrightarrow{\omega_k^m} S_{ik+1}^m$
    Ermittle $c_{ik+1}^{det}(S_{ik}, d_{ik}, \omega_{ik})$.
  end
  Erhöhe m.
end
return $\overline{V}^M(S_{ik}^{d_{ik}})$, $\forall S_{ik}^{d_{ik}}$

troffen werden. Diese wird zu jedem Entscheidungszeitpunkt k in einem Vektor $(d_{1k}, ..., d_{nk})$ zusammengefasst und dem Steuerungslevel zur Umsetzung in der Simulation übergeben. Dabei läuft die Schätzung der sofortigen Repositionierungskosten $\hat{c}_{ik}^{tr}$ in einem parallelen ADP-Verfahren. $(\hat{c}_i^{tr})^0$ wird mit der Entfernung $d(0, i)$ zum Depot initialisiert. Sei μ der Parameter, der zählt, wie oft eine Station i in eine Repositionierungstour eingebunden wird. Das Update erfolgt immer dann, wenn die entsprechende Station in der aktuellen Repositionierungstour eingeplant ist, nach der Formel

$$\left(\hat{c}_i^{tr}\right)^{\mu} = (1 - \alpha_{\mu-1}) \cdot \left(\hat{c}_i^{tr}\right)^{\mu-1} + \alpha_{\mu-1} \cdot \left(\overline{c}_i^{tr}\right)^{\mu}. \tag{5.6}$$

Dabei ist $(\overline{c}_i^{tr})^{\mu}$ der aktuelle Kostenzuwachs, der in der Tour entsteht, wenn Station i hinzugefügt wird. Er berechnet sich durch

$$\left(\overline{c}_i^{tr}\right)^{\mu} = d(i-1, i) + d(i, i+1) - d(i-1, i+1), \tag{5.7}$$

wie in Abbildung 5.3 verdeutlicht wird. $(\hat{c}_i^{tr})^{\mu-1}$ ist der vorhergehende Schätzwert und α hat wieder eine glättende Aufgabe.

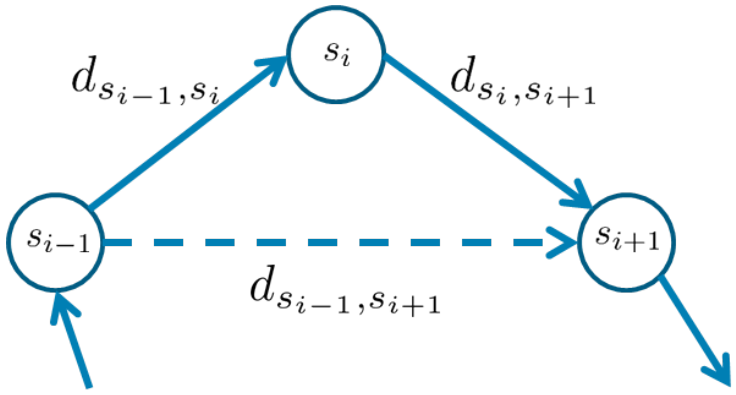

Abbildung 5.3.: Kostenzuwachs $(\overline{c}_i^{tr})^n$

Wenn die gewählte Anzahl an Trajektorien M erreicht ist, erhalten wir die finale lookup-table $\left(\overline{V}^M(S_{ik}^{d_{ik}})\right)$, welche nun in Testsettings angewendet und ausgewertet werden kann. Wie genau die verschiedenen Testsettings in dieser Arbeit aufgebaut sind, wird in Kapitel 6 erläutert.

6. Experimente

In diesem Kapitel werden das mathematische Modell und die Methodik aus den Kapiteln 3 und 5 umgesetzt, auf realen sowie künstlich erzeugten Daten getestet und mit einfachen, intuitiven Verfahren vergleichen. Kapitel 6.1 erläutert den Aufbau der Experimente, während in Kapitel 6.2 verschiedene Entscheidungspolitiken beschrieben werden. Kapitel 6.3 stellt die Daten vor, die zur Simulation verwendet werden und in Kapitel 6.4 werden die Ergebnisse gezeigt.

6.1. Aufbau

Als Grundlage unseres experimentellen Aufbaus dient die in Kapitel 5.1 beschriebene Simulation. Für die Untersuchungen werden *fünf Tage* à 24 Stunden kontinuierlich simuliert ($T = 5 \cdot 24, t \in [0, T]$). Die Entscheidungspunkte $k = 1, ..., K$ liegen *äquidistant* in T. Jeweils nach *zwei Stunden* Simulation kann über eine Repositionierung entschieden werden. Für jedes System wird ein Depot $0 \in V$ angenommen, das innerhalb des Systembereichs liegt. Dort befinden sich die Repositionierungstransporter zu Beginn des Simulationshorizontes. Die Anzahl $|V|$ der vorhandenen Transporter ist variabel und wird je nach Systemgröße und Nachfragestärke angepasst. Es wird aber von einer festen Ladekapazität $Q_v = 20$ ausgegangen. Während der Simulation werden Bewegungsdaten im System aufgezeichnet. Darunter fallen zum Beispiel die Anzahlen der erfolgreichen oder nicht erfolgreichen Entleihungen und Rückgaben sowie der Repositionierungstouren und der repositionierten Räder. Wichtig ist zudem die Speicherung der Kundenumwege und die Wege und Umwege sowie Lade- und Abladezeiten der Transportfahrzeuge. Diese dienen zur Bewertung der Repositionierungsperformance als Bestandteil der Zielfunktion. Durch einen Gewichtungsparameter w kann die Präferenz zur Bevorzugung oder Vernachlässigung des kundenseitigen Aufwandes gegenüber dem operativen Aufwand variiert werden. Der Repositionierungsaufwand setzt sich aus den gefahrenen Touren und der Zeit zum Auf- und Abladen der Räder zusammen. Den gefahrenen Strecken liegen, wie auch den Kundenumwegen, Google-Fahrtzeiten zugrunde. Im

Systemablauf kann es vorkommen, dass eine geplante Tour wegen unerwarteten Kundenbewegungen nicht durchführbar ist. Sollte ein Transporter zum Ende der Tour noch Räder aufgeladen haben, so muss er Umwege fahren, um diese an anderen Stationen zu entladen. Sollte er keine Räder mehr geladen haben und keine Angebotsstation mehr vor sich haben, so kann die Tour auch kürzer ausfallen. Für das Ergebnis der Simulation zählt die tatsächlich gefahrene Strecke sowie die tatsächlich aufgewendete Auf- und Abladezeit. Für die Ladezeit wird dabei eine konstante Zeit von einer Minute pro Rad angenommen.

6.2. Entscheidungspolitiken

Um den in dieser Arbeit vorgeschlagenen Ansatz mittels ADP zu evaluieren, vergleichen wir ihn mit zwei myopischen Ansätzen. Im folgenden sind die Settings aller drei Methoden zusammengestellt.

6.2.1. Greedy bzw. Nichts-Tun-Strategie

Die Greedy-Methode vermeidet jegliche Transportkosten, da sie keine zukünftigen Strafkosten berücksichtigt. Sie ist damit gleichzusetzen, dass ein Systembetreiber, das System auf sich allein stellt. Es finden keine betreiberseitigen Repositionierungen statt.

6.2.2. Myopische reaktive Strategie

Der myopische reaktive Ansatz benötigt als Eingabeparameter relative Puffergrenzen und Repositionierungsmengen. In dieser Arbeit werden Puffergrenzen von 0% bis 30% verglichen. Die Repositionierungsmenge ist von der Größe des Entscheidungsraumes abhängig und flexibel einstellbar. Hier wurde eine Menge entsprechend der Aggregationsstufen der Füllstände gewählt. Eine Nachfrage nach Rädern oder Stellplätzen erfolgt genau dann an einer Station, wenn die Puffergrenzen erreicht oder unter- bzw. überschritten werden. Im Falle von 0% tritt die Reaktion erst ein, wenn die Station voll oder leer ist. Im Falle von 20% wird die Reaktion bei einem Bestand unter 20% oder über 80% ausgelöst.
Der resultierende Nachfragevektor ist mit hoher Wahrscheinlichkeit nicht ausgeglichen, d.h. die Summe der Nachfrage nach Rädern ist ungleich der Summe der Nachfrage nach Stellplätzen. Da ein Transporter auf der Tour weder Räder erzeugen noch verschwinden lassen kann, muss der Vektor noch ausgeglichen werden. Wie

in Kapitel 5.3 erläutert, wurde dafür die Funktion `findBestFeasible(demand)` implementiert. Der resultierende ausgeglichene Nachfragevektor wird an die Tourenplanung (abstrakte Klasse `RoutePlanner`) weitergegeben, die daraus eine Tourenliste erstellt.

6.2.3. Antizipierende Strategie

Wie in Kapitel 5.4 beschrieben, brauchen wir für den ADP Ansatz eine Reihe von Parametern, die die Optimierung entscheidend beeinflussen. Als erstes sind die Aggregationsparameter festzulegen. Die Anzahl der Zustände `numStates` ist in den Experimenten auf 5 für jede Station gesetzt ($S_{ik} \in \{0, ..., 4\}$, $\forall i$). Die Anzahl der Entscheidungen `numDecisions` ist 3, mit $d_{ik} \in \{-1, 0, 1\}$. Dabei entspricht $d_{ik} = 1$ einer Lieferung an Rädern, während $d_{ik} = -1$ für eine Abholung steht. $d_{ik} = 0$ bedeutet, dass keine Aktion an Station i zum Entscheidungszeitpunkt k stattfindet. Als nächstes werden die Parameter des Lern- bzw. Updateprozess benötigt. Die Anzahl der Trajektorien M legt fest, wie lange der Lernprozess andauert. Der Diskontierungsfaktor γ bestimmt die Gewichtung der zukünftigen Ereignisse für deren Einfluss in die aktuelle Entscheidung. M und γ gehören zu den regulierbaren Parametern, die in Kap. 6.2.3.1 zusammengestellt sind. Der Schrittweitenparameter α aus der Updategleichung

$$\overline{V}^{d_{ik}m_{ik}}(S_{ik-1}^{d_{ik}}) = (1 - \alpha_{m_{ik}-1})\overline{V}^{d_{ik}m_{ik}-1}(S_{ik-1}^{d_{ik}}) + \alpha_{m_{ik}-1}\hat{v}_{ik}^{m_{ik}} \tag{6.1}$$

wird mit $\alpha_{m_{ik}} = \frac{1}{m_{ik}+1}$ festgelegt, wobei m_{ik} ein Counter ist, der die Anzahl der Updates für jeden Zustand zählt. So fließt das Resultat jeder Aktion zu gleichen Teilen in das Wissen über den Zustand $S_{ik}^{d_{ik}}$ ein. Die Werte $V^0(S_{ik}^{d_{ik}})$ werden mit 0 initiiert.

Mit diesen Parametern kann das ADP-Verfahren gestartet werden. Jede Trajektorie nutzt einen Simulationsdurchlauf, um nach den jeweils aktuellen Werten $\overline{V}$ zu optimieren. Die Performance der einzelnen Entscheidungen bzgl. der Zielfunktion aus Formel 3.17 fließen in die Updates der Zustandswerte ein. Nach den M Trajektorien stoppt der Lernprozess und wir erhalten eine Zustandswertetabelle, nach der in jedem Entscheidungszeitpunkt über die Repositionierungsaktionen an allen Stationen entschieden werden kann. Dies erfolgt nach

$$d_{ik}^{m*} = \underset{d_{ik} \in \mathcal{D}_i}{\operatorname{argmin}} \left(\hat{c}_{ik}^{tr}(d_{ik}) + \gamma \overline{V} \left(S_{ik}^{d_{ik}} \right) \right). \tag{6.2}$$

Ist dies für alle Stationen i erfolgt, erhalten wir einen Nachfragevektor, der dem Vektor aus der reaktiven Strategie entspricht. Auch hier besteht in den meisten Fällen Ausgleichsbedarf, sodass die Summe der Stellplatznachfragen gleich der Summe der Radnachfragen ist. Analog zum Vorgehen der myopischen Strategie wird eine angepasste Methode `findBestFeasible(demand)` implementiert. Diese hat gegenüber der ursprünglichen Version den Vorteil, dass sie iterativ die Transporte mindert oder streicht, die den geringsten Wertzuwachs erzeugen. Es fließen also die gelernten Werte in die Entscheidung ein. Die Umsetzung in Touren erfolgt wiederum nach Kapitel 5.3.

6.2.3.1. Regulierbare Parameter

Gewichtungsparameter w für Transport- bzw. Umwegaufwand: Um die Präferenzen verschiedener Systembetreiber in die Optimierung einfließen lassen zu können, existiert ein Gewichtungsparameter w der die beiden konkurrierenden Ziele im Lernprozess verschieden berücksichtigen kann. Aus w werden innerhalb der Simulation zwei Faktoren `factorForTransportationCost` (w^{tr}) und `factorForDetourCost` (w^{det}) abgeleitet. Diese beeinflussen dann jeweils die Auswahl der optimalen Aktion und die Addition der Umwege auf den aktuellen Wert der Trajektorie.

$$\hat{v}_{ik}^{m} = w^{det} \cdot c_{ik}^{det}(S_{ik-1}, d_{ik-1}, \omega_{ik-1}) + \min_{d_{ik} \in \mathcal{D}_i} \left(w^{tr} \cdot \hat{c}_{ik}^{tr}(d_{ik}) + \gamma \overline{V} \left(S_{ik}^{d_{ik}} \right) \right) \tag{6.3}$$

Die tatsächlich anfallenden Zeiteinheiten bleiben aber bestehen und werden nicht verzerrt.

Anzahl der Trajektorien M: Mit jeder Trajektorie $m \in M$ wächst das Wissen über die in m besuchten Zustände. Mit einem dämpfenden Schrittweitenparameter α nähert sich die Lernkurve üblicherweise für steigende $m \to \infty$ einer Asymptote an. Je nach Anzahl der Zustände insgesamt ist eine kleinere oder größere Anzahl für das gewünschte Lernergebnis angemessen. Beim vorliegenden Grad der Aggregation sowie der Dekomposition des Problems, erreicht die Lernkurve schon nach wenigen 1000 Trajektorien diese Asymptote.

Diskontierungsfaktor γ: γ bestimmt den Einfluss der Zukunft auf die aktuelle Entscheidung. $\gamma = 1$ würde bedeuten dass die zukünftig möglichen Ereignisse

mit den jetzigen gleichgesetzt werden, während $\gamma = 0$ die Zukunft gar nicht mit einbezieht. Welche Größe von γ für die jeweilige Optimierung passend ist, hängt von der Stochastizität des Problems ab. Sind die Zufallsausgänge dicht beieinander und treten gehäuft ähnliche Zustände nach einem bestimmten Ausgangszustand auf, so hilft ein hohes γ der Lerngeschwindigkeit. Sind die Zufallsausgänge allerdings weit gestreut und treten vom gleichen Ausgangszustand immer wieder sehr verschiedene Folgezustände auf, so hilft ein niedriges γ das Optimierungsverfahren nicht „überreagieren“ zu lassen.

Anzahl der Repositionierungstransporter $|V|$: Je nach Größe des BSS ist eine kleinere oder größere Anzahl von Transportern notwendig um den Repositionierungsbedarf zu decken. Sind zu wenige Transporter vorhanden, so kann eine geplante Aktion evtl. nicht ausgeführt werden. Hier ist darauf zu achten, dass der Vergleichbarkeit wegen im myopischen sowie antizipierenden Ansatz die gleiche Anzahl an Transportern zur Verfügung stehen muss.

6.3. Inputdaten

Für die Experimente auf Praxisdaten, liegen Daten des CityBike in Wien aus den Jahren 2008 und 2009 vor. Es werden Daten auf zwei verschiedene Arten bereitgestellt: Bei einer Art handelt es sich um sogenannte Stationssnapshots, bei der anderen um Kundenfahrtdaten. Die Tabelle der Snapshotdaten enthält ca. alle fünf Minuten zu jeder Station einen Eintrag mit eindeutiger ID, einem Zeitstempel und der jeweiligen Stationsnummer sowie der Stationskapazität. Festgehalten wird zu jedem Snapshotzeitpunkt jeweils die Anzahl an funktionierenden, gestörten, oder gesperrten Stellplätzen, die Anzahl an freien Rädern bzw. freien Stellplätzen und die Anzahl an beschädigten Rädern. Eine beispielhafte Zeile mit den wichtigsten Einträgen ist in Tabelle 6.1 abgebildet.

Tabelle 6.1.: Stationssnapshots

stationsnapshot id	zeitpunkt	station	boxen count	boxen ok	boxen bike	boxen bike empty
13359981	“2009-12-30 17:47:11“	1022	36	35	17	18

Tabelle 6.2.: Fahrtdaten

ride id	entleih-station	entleih-zeitpunkt	rueckgabe-station	rückgabe-zeitpunkt	entleihdauer
579248	1021	“2009-06-10 18:06:16“	1060	“2009-06-10 18:25:33“	1157

Die Kundenfahrtdaten enthalten Informationen über Entleih- bzw. Rückgabezeit und -ort einer jeden Fahrt, die im System im Aufzeichnungszeitraum stattgefunden haben. Zudem sind auch Informationen über den Kunden (beispielsweise der Wohnort) gespeichert, die zu unserem Zweck nicht verwendet wurden. Die wichtigen Spalten der Fahrtdaten sind an einer Beispielzeile in Tabelle 6.2 dargestellt. Mit diesem umfangreichen Datensätzen als Basis können nun die Inputdatentabellen für unsere Experimente generiert werden. Der Aufbau des Simulationsprogramms fordert vier Tabellen:

- Stationsdaten mit Informationen über Name, Position und Kapazität der Stationen
- Fahrtzeiten von jeder Station zu jeder Station
- Nachfrageparameter λ für jede Station zu jeder Stunde eines Werktages
- Verteilung der Zielstationen für jede Station zu jeder Stunde eines Werktages

Mittels Datenbankabfragen, Datenanalyse und entsprechenden Aggregationen werden diese aus den Rohdaten von CityBike Wien gewonnen und dem Simulationsprogramm als .csv-Dateien zur Verfügung gestellt.
Eine umfangreiche Datenanalyse auf Basis dieser Daten wird von Vogel et al. durchgeführt und ist in [80] nachzulesen. Basierend auf den Ergebnissen dieser und folgender Arbeiten entwickelt Vogel [79] ein Verfahren, welches künstliche BSS in beliebiger Größe mit variablen Nachfrageparametern simulieren kann. Dabei werden übliche Cluster von Stationstypen generiert und abhängig davon die Flussstärken von Kundennachfragen zwischen den Stationen festgelegt. Für die Experimente auf künstlichen Daten wurden Instanzen nach diesem Verfahren generiert. Sie bestehen jeweils aus zwei unterschiedlichen Stationstypen. Durch die Variation der Nachfragestärken- und Systemgrößenparameter kann dann untersucht werden, für welche Art von Systemen welche Art von Repositionierungsstrategien anwendbar und anpassbar ist.

Die folgenden Unterkapitel stellen die Ergebnisse der Experimente zusammen. Dabei bezieht sich Kapitel 6.4.1 auf die Praxisdaten, während Kapitel 6.4.2 die Ergebnisse der Experimente auf künstlichen Daten enthält.

6.4. Ergebnisse

Um die Ergebnisse der einzelnen Verfahren zu vergleichen, wird die Simulation für jede Parameterkombination 100 mal durchlaufen. Gespeichert werden jeweils die Transport- und Kundenumwege, die Servicezeiten zum Be- und Entladen der Transporter, die erfolgreichen und nicht erfolgreichen Entleihungen und Rückgaben, sowie Anzahl der Touren und der repositionierten Räder. Zum Vergleich herangezogen werden dabei die Durchschnitte aus den 100 Simulationen.

6.4.1. Praxisdaten

Erwartungsgemäß führt die Greedy-Strategie ohne Repositionierungsaufwand zu sehr hohen Umwegkosten für die Kunden. Der Abbildung 6.3 können die Umwege in Stunden pro Tag sowie die Anzahl und der Prozentsatz der fehlgeschlagenen Nachfragen entnommen werden. Von den Nachfragen nach Rädern werden knapp 56% nicht erfüllt. Von den Nachfragen nach Stellplätzen sind es fast 28%. Diese schlechte Quote führt zu Kundenumwegen von insgesamt $117,8$ Stunden pro Tag. Es steht also außer Frage, dass ein operatives Eingreifen durch Repositionierung unumgänglich ist.

Tabelle 6.3.: Ergebnisse bei Greedy-Strategie

	Zeit$[\frac{h}{Tag}]$		**Anzahl (%) fehlender**	
	c^{det}_{walk}	c^{det}_{bike}	Räder	Boxen
Greedy	$90,0$	$27,8$	$983,6\ (55,9)$	$490,8\ (27,9)$

Der Effekt von Repositionierungsaktionen wird mit einem Blick auf die Tabelle 6.4 ersichtlich. Mit der myopischen Strategie ohne Puffer kann durch 2 Transporter mit einem Transport- und Serviceaufwand von ca. 16 Stunden der Kundenumweg insgesamt um über 100 Stunden reduziert werden. Bei steigendem Puffer wächst der operative Aufwand an, während die Umwege zunächst ein wenig weiter sinken. Bei einem Puffer von 30% werden die Umwegzeiten allerdings wieder größer. Das liegt daran, dass zu früh reagiert wird. Füllstände nahe der Kapazitätsgrenzen, die

für eine Station günstig sein könnten, werden bei Über- oder Unterschreitung des Puffers verändert. Dies kann zu entgegengesetzten Fehlmengen führen. Auch bei kleineren Puffern kann es zu Fehlern durch zu frühes Reagieren kommen. Aus diesem Grund und da das Wachstum des Transport- und Serviceaufwands verglichen mit der Senkung des Umwegs zu hoch ausfällt, wird zu weiteren Vergleichen der myopische Ansatz ohne Puffer herangezogen.

Tabelle 6.4.: Myopische Strategien im Vergleich

	Zeit$[\frac{h}{Tag}]$				**Anzahl**	
	c^{tr}	c^{s}	c^{det}_{walk}	c^{det}_{bike}	Touren	repos. Räder
0%	$11,7$	$4,5$	$9,8$	$6,0$	$12,9$	$135,5$
10%	$15,5$	$6,3$	$6,3$	$3,3$	$16,2$	$190,5$
20%	$18,8$	$8,2$	$5,5$	$2,9$	$19,3$	$246,5$
30%	$20,2$	$9,4$	$8,1$	$3,4$	$20,3$	$281,2$

Für das ADP-Setting ist die Anzahl der Transporter, wie auch im Experiment zum myopischen Ansatz, zwei. Das ist die Anzahl an Transportern, die im Wiener BSS zur Zeit der Datenaufzeichnung im Einsatz waren. Das Verhältnis von Transport- zu Umwegzeit wird $1:1$ ($w=1$) gesetzt. γ ist hier $0,5$. Für die Lernphase werden 5000 Trajektorien durchlaufen.
In Abbildung 6.1 sind die Komponenten der Zielfunktion, Transport- und Servicezeiten auf Betreiberseite sowie die Umwege auf Kundenseite, für die ersten 500 Trajektorien abgetragen. Die schnelle Konvergenz der einzelnen Kurven ist gut zu erkennen. Begründen kann man sie durch die geringe Anzahl der Zustände an den einzelnen Stationen sowie der multiplen Updates der Zustandswerte pro Trajektorie. Da jede Trajektorie eine Länge von fünf Tagen hat, werden auch fünf Updates von Werten zu jedem Entscheidungspunkt eines Tages durchgeführt. Während Transport- und Servicezeit sich ihrer Asymptoten von unten nähern, fallen die Umwege für die Kunden zu Anfang steil ab, um sich von oben der Asymptoten zu nähern. Der Anstieg der Transport- und Servicekurven begründet sich in der Initialisierung der Anfangswerte der Zustände. Da alle Zustände zunächst gleich bewertet sind, werden Repositionierungsaktionen vermieden, da diese zusätzlichen Zeitaufwand bedeuten. Schon nach wenigen Durchläufen häufen sich jedoch die Strafkosten an und die entsprechenden Zustände erhalten eine schlechtere Bewertung. Um diese dann zu vermeiden, werden Repositionierungen ausgelöst. Diese führen dazu, dass die Umwegzeiten stark zurückgehen. Das Zittern der Kurven

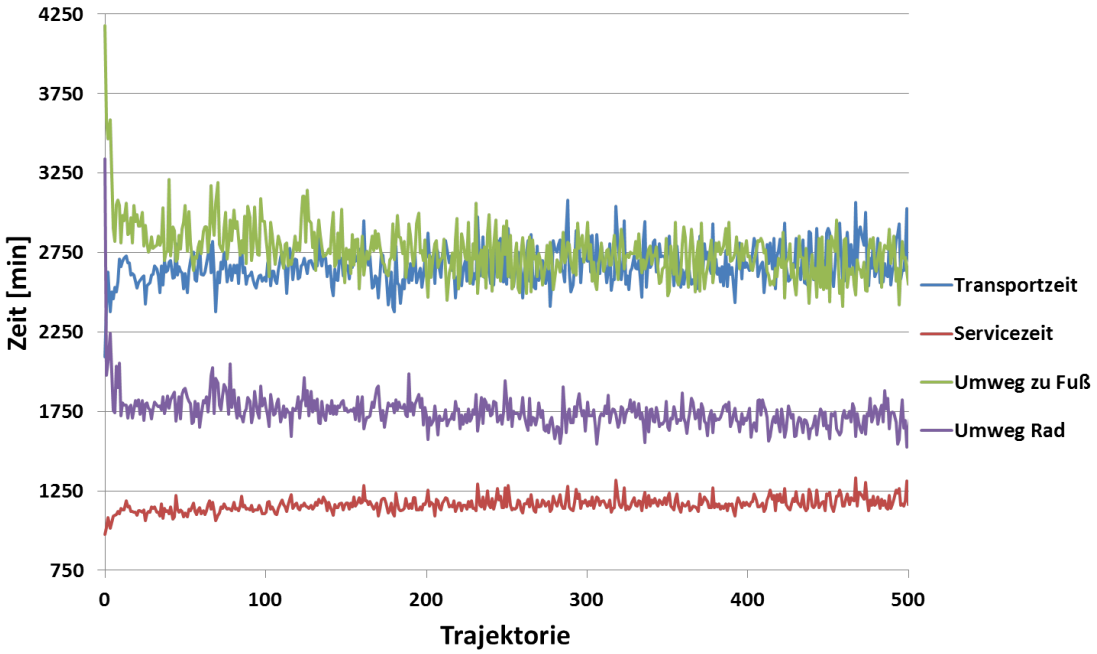

Abbildung 6.1.: Komponenten der Zielfunktion im Lernverlauf

ist durch die Stochastizität der einzelnen Trajektorien zu erklären. Aufsummiert ergeben die vier Kurven den Wert der verwendeten Zielfunktion. Deren Lernkurve ist in Abbildung 6.2 abgetragen. Durch die Addition ist die schnelle Konvergenz noch deutlicher zu erkennen und das Zittern ist stärker ausgeprägt.
In der Tabelle 6.5 wird der Vergleich zwischen der myopischen Strategie ohne Puffer mit dem ADP-Ansatz gezogen.

Tabelle 6.5.: ADP im Vergleich mit der myopisch reaktiven Strategie

	Zeit$[\frac{h}{Tag}]$				**Anzahl**			
	c^{tr}	c^{s}	c^{det}_{walk}	c^{det}_{bike}	fehlende Räder	fehlende Boxen	Touren	repos. bikes
myopisch (Puffer 0%)	11,7	4,5	9,8	6,0	114,9 (6,5%)	113,8 (6,5%)	12,9	135,5
ADP	9,4	4,1	8,4	5,4	116,8 (6,6%)	115,7 (6,6%)	12,5	123,0

In allen Komponenten der Zielfunktion ist die durch den ADP-Ansatz erreichte Zeit geringer als im myopischen Ansatz. Es werden vergleichbar viele Touren geplant, jedoch sind diese effektiver. Die Anzahl der fehlgeschlagenen Entleihungen und Rückgaben ist geringfügig höher. Das liegt an der Beschaffenheit der Zielfunktion. An Stationen, die einen sehr geringen Umweg erzeugen, etwa in dem Fall, wenn die

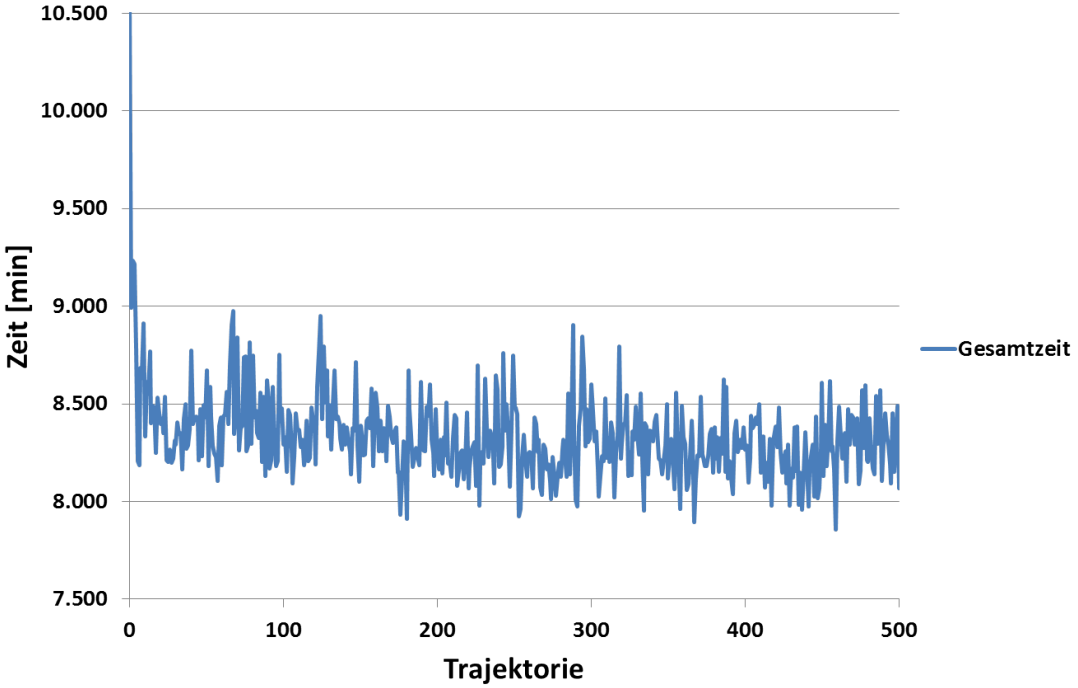

Abbildung 6.2.: Wert der Zielfunktion im Lernverlauf

nächste Station in Sichtweite liegt, sind im Vergleich weniger schwerwiegend. Diese Argumentation kann nur standhalten, wenn die Kundenunzufriedenheit proportional zum Umweg steigt. Falls dem nicht so ist, kann dem Modell ein Strafterm pro fehlgeschlagener Serviceleistung hinzugefügt werden. Von dieser Variante wurde hier abgesehen, da es eine gewisse Willkür mit sich bringt, diesen Strafterm zu bewerten. Liegen dem Anwender diese Daten jedoch vor, steht einer Veränderung der Zielfunktion nichts im Weg. Je nach Zielsetzung können auch gänzlich andere oder eine andere Zusammensetzung an Zielfunktionen aufgestellt werden.

6.4.2. Künstliche Daten

Für die Experimente auf künstlichen Daten wurden Instanzen kleiner (36 Stationen) und mittelgroßer (100 Stationen) Systemgrößen gewählt. Die Stationen sind gitterförmig angeordnet und das Depot befindet sich nahe der Mitte. Dabei stehen im 36-Stationen-System 2 Repositionierungsfahrzeuge zur Verfügung und im 100-Stationen-System 8. Die Stationskapazitäten wurden entweder durchgängig auf 30 Stellplätze oder auf 60 Stellplätze gesetzt. Die Belegung der Stellplätze ist mit 50% initialisiert. Die durchschnittliche Nachfrage wurde jeweils mit einer, 2, oder 4 Fahrten pro Rad und Tag im System mit 60 Stellplätzen pro Station festgesetzt.

Im System mit 30 Stellplätzen pro Station ist die Nachfrage pro Rad und Tag bei gleichbleibender Gesamtnachfrage folglich doppelt so hoch. Damit wurde das unterschiedliche Verhalten von Systemen mit geringer, mittlerer und hoher Nachfrage untersucht. Das System basierend auf den Wiener Daten hat eine Nachfrage pro Rad und Tag von ca. 3. Tabelle 6.6 zeigt eine Übersicht der Nachfragegrößen in den verschiedenen Instanzen. Die Anzahl der Nachfragen pro Tag sind Durchschnittswerte aus 100 Läufen über 5 Tage. Die Inputtabellen werden aus diesen Angaben und den entsprechenden Informationen über die Stationstypen erzeugt und in dieselbe Form wie bei den Praxisexperimenten gebracht, um sie in gleicher Weise verarbeiten zu können.

Tabelle 6.6.: Nachfrageverhalten in den verschiedenen künstlichen Instanzen

Anzahl		**Nachfrage**	**Anzahl Entleihungen/Rückgaben**	
Stationen	Boxen		pro Tag	pro Tag und Rad
36	30	niedrig	1080	2
		mittel	2161	4
		hoch	4323	8
	60	niedrig	1080	1
		mittel	2161	2
		hoch	4323	4
100	30	niedrig	2997	2
		mittel	6000	4
		hoch	12005	8
	60	niedrig	2997	1
		mittel	6000	2
		hoch	12005	4

Wird die Greedy-Politik angewendet, so kann man, wie auch bei den Praxisexperimenten, das pure Systemverhalten ohne Eingriff durch Repositionierungen ablesen. In Tabelle 6.7 ist die jeweilige Serviceperformance zum Vergleich der verschiedenen künstlichen Instanzen aufgetragen. Die Intuition legt nahe, dass bei steigender Nachfrage sowie Verringerung der Stationsgröße die Kundenumwege ansteigen. Dies wird mit den Einträgen in der Tabelle belegt.
Mit dem Wissen über die Ergebnisse der Experimente auf Praxisdaten würden ohne Repositionierungsmaßnahmen erhebliche Kundenumwege erwartet. Tatsächlich treten zwar Umwege auf, aber in einem deutlich geringerem Maß. Während wir in der Simulation des BSS in Wien noch über 50% fehlgeschlagene Entleihungen und mehr als ein Viertel fehlgeschlagene Rückgaben beobachten (siehe Tabelle

Tabelle 6.7.: Performance der Greedy-Strategie in den verschiedenen künstlichen Instanzen

Greedy			**Zeit**$[\frac{h}{Tag}]$		**Anzahl (%)**	
Anzahl		Nachfrage	c^{det}_{walk}	c^{det}_{bike}	fehlende Räder	fehlende Boxen
Stationen	Boxen					
36	30	niedrig	12,6	10,5	79,5 (7)	98,8 (9)
		mittel	62,7	24,8	394,2 (18)	233,5 (11)
		hoch	123,2	45,4	774,8 (18)	427,8 (10)
	60	niedrig	2,1	2,7	13,3 (1)	25,1 (2)
		mittel	20,4	11,3	128,1 (6)	106,7 (5)
		hoch	51,6	28,4	324,6 (8)	267,4 (6)
100	30	niedrig	31,9	16,2	335,8 (11)	265,1 (9)
		mittel	86,2	41,6	905,8 (15)	655,7 (11)
		hoch	264,3	100,2	2779,2 (23)	1578,8 (13)
	60	niedrig	6,3	2,9	66,7 (2)	45,7 (2)
		mittel	26,0	15,2	273,7 (5)	239,6 (4)
		hoch	114,8	56,4	1208,1 (10)	890,3 (7)

6.3), sind es in den künstlichen Systemen maximal 23% und 13%, bei der höchsten Nachfrage und der kleinsten Stationsgröße. Dies ist auf das Systemdesign zurückzuführen. Die künstlichen Systeme beruhen auf gemittelten Daten und bestehen aus nur zwei verschiedenen Stationstypen (Vogel et al. fanden im Wiener BSS fünf verschiedene Stationstypen mit unterschiedlichen Entleih- und Rückgabespitzen [80]). Trotz der im Vergleich knapp dreifachen Nachfrage scheint es sich besser selbst balancieren zu können. Das liegt an den fehlenden Überlagerungen der Effekte weiterer Stationstypen. Im Vergleich zu den realen Daten sind hier nur die Nachfrageverhalten von Pendlern berücksichtigt worden. So gibt es nur verstärkte Flüsse zur Morgenspitze von einem zum anderen Stationstyp und zur Abendspitze in die entgegengesetzte Richtung. Touristen und nachtaktive Nutzer wurden außer Acht gelassen. Dies führt dazu, dass sowohl die Nacht- als auch die Tagaktivität geringer ausfällt als in Systemen wie dem CityBike Wien. Zudem sind durch die Gleichförmigkeit der Stationskapazitäten keine außerordentlich kleinen Stationen mehr vorhanden, welche durch ihre geringen Puffer eher dazu neigen an ihre Kapazitätsgrenzen zu stoßen. Zusammengefasst ist ein geringerer Repositionierungsbedarf vorhanden, was weniger Potential zur Optimierung lässt. Myopischen Strategien kommen jedoch Systeme mit wenig Repositionierungsbedarf entgegen.

Die verzögerte Reaktion fällt weniger ins Gewicht.
Tabelle 6.8 fasst die Ergebnisse der Anwendung unserer myopischen Repositionierungspolitik auf den verschiedenen Instanzen zusammen. Mit vergleichbar kleinem Transport- und Serviceaufwand werden die Kundenumwege deutlich reduziert. In den Instanzen mit geringer Nachfrage gehen die fehlgeschlagenen Entleihungen und Rückgaben sogar auf unter ein halbes Prozent zurück.
Die Ergebnisse unter Verwendung der antizipierenden Strategie liegen in etwa der gleichen Größenordnung wie unter Verwendung der myopischen Strategie. Jedoch ist sie nicht so dominant, wie in den Experimenten mit den Praxisdaten aus Wien. Für jede der Testinstanzen wurde die Parameterkombination gewählt, die den kleinsten Zielfunktionswert erzeugt. Eine Übersicht über die gewählten Werte ist in Tabelle 6.9 gegeben.
Es fällt auf, dass sich die Parameterkombinationen bei gleich hoher Nachfrage ähneln. So ist in drei von vier Fällen jeweils die Gewichtung der Zeiten auf Kunden- und Betreiberseite gleich und γ liegt in ähnlichen Größenordnungen. Bei hoher Nachfrage scheint ein größeres γ zu besseren Ergebnissen zu führen, während bei niedriger Nachfrage Werte um $0,6$ herum besser abschneiden. Das lässt darauf schließen, dass die Nachfragemuster bei höherer Nachfrage besser ausgeprägt sind, sodass eine stärkere Einbeziehung der Zukunft vorteilhaft sein könnte. Die nachfolgenden Tabellen 6.10, 6.11 und 6.12 vergleichen jeweils die beiden Strategien in allen Instanzen, aufgeteilt nach der Höhe der Nachfrage. Dabei stellt sich ein Vorteil der myopischen Strategie deutlich heraus: Die Reaktion findet nur bei Erreichung eines bestimmten Füllstandes (voll oder leer) statt. Die antizipierende Strategie lernt allerdings ihre Reaktionen auf den grobkörnigen Füllstandsintervallen. Eine leere Station hat folglich denselben Wert, wie eine bis zum Ende des ersten Füllstandsintervall gefüllte. Dies kann besonders dann zu falschen Entscheidungen führen, wenn durch eine geringe Anzahl an Bewegungen, oder einen hohen Selbstausgleich, kleine Veränderung im Füllstand beträchtlichen Einfluss haben.

Bei niedriger Nachfrage (siehe Tabelle 6.10) schneidet der ADP-Ansatz bezüglich der Zielfunktion in allen Fällen schlechter ab. Das liegt daran, dass bei den wenigen Bewegungen im System auch nur wenige Kundenumwege auftreten. Diese sind wegen der stochastischen Fahrradflüsse systemweit verteilt und lassen sich nicht vorhersehen. Der myopische Ansatz hat den Vorteil, dass er immer im letzten Augenblick an der richtigen Station reagiert und wegen der niedrigen Nachfrage dafür auch mehr Zeit hat. Der ADP-Ansatz lernt durch die sporadisch auftretenden Kun-

Tabelle 6.8.: Performance der myopischen Strategie in den verschiedenen künstlichen Instanzen

myopisch (Puffer)			**Zeit**[$\frac{h}{Tag}$]				**Anzahl**			
Anzahl		Nachfrage	c^{tr}	c^{s}	c^{det}_{walk}	c^{det}_{bike}	fehlende Räder	fehlende Boxen	Touren	repos. Räder
Stationen	Boxen									
36	30	niedrig	2,9	1,3	2,6	1,4	16,5 (1,5%)	13,2 (1,2%)	5,0	39,8
		mittel	5,4	2,6	15,0	5,0	94,2 (4,4%)	47,5 (2,2%)	7,5	76,6
		hoch	7,3	3,6	51,7	10,9	325,1 (7,5%)	103,1 (2,4%)	9,3	108,5
	60	niedrig	0,9	0,6	0,4	0,4	2,8 (0,3%)	3,6 (0,3%)	1,5	18,4
		mittel	2,9	1,8	3,4	1,9	21,4 (1,0%)	17,9 (0,8%)	4,1	54,8
		hoch	5,0	3,1	16,0	5,2	100,4 (2,3%)	48,8 (1,1%)	6,7	92,9
100	30	niedrig	7,1	3,6	6,4	2,3	67,1 (2,2%)	37,0 (1,2%)	8,9	108,1
		mittel	12,1	6,6	24,6	8,6	259,3 (4,3%)	135,7 (2,3%)	13,4	198,0
		hoch	17,6	10,3	106,3	25,2	1118,6 (9,3%)	397,6 (3,3%)	18,0	307,8
	60	niedrig	2,7	1,7	1,1	0,5	11,3 (0,4%)	8,3 (0,3%)	3,9	52,1
		mittel	7,8	4,7	5,7	3,2	59,6 (1,0%)	50,4 (0,8%)	9,6	142,0
		hoch	15,7	9,4	31,3	12,0	329,7 (2,7%)	189,9 (1,6%)	17,5	281,0

Tabelle 6.9.: Parameterkombination mit bester Zielfunktion

Anzahl		**Nachfrage**	**Parameter**	
Stationen	Boxen		w	γ
36	30	niedrig	1 : 1	0, 69
		mittel	2 : 1	0, 83
		hoch	3 : 1	0,99
	60	niedrig	1 : 1	0, 57
		mittel	2 : 1	0, 90
		hoch	3 : 1	0,85
100	30	niedrig	1 : 1	0, 63
		mittel	2 : 1	0, 89
		hoch	2 : 1	1, 00
	60	niedrig	2 : 1	0, 83
		mittel	4 : 1	0, 74
		hoch	3 : 1	0, 92

denumwege und die starke Aggregation zu früh zu reagieren und transportiert so evtl. unnötig Räder hin und her. Dies ist im Fall von 36 Stationen mit 30 Stellplätzen gut zu erkennen. Trotz einem höheren Zeitaufwand für Repositionierungen fallen die Kundenumwege höher aus. Bei 60 Stellplätzen, oder dem größeren System mit 100 Stationen treten so wenige Umwege auf, dass der ADP-Algorithmus ihnen nicht ausreichen Beachtung schenkt. Es werden folglich weniger Repositionierungen ausgelöst, aber dafür mehr Umwege in Kauf genommen. Insgesamt führt der ADP-Ansatz so zu einem schlechteren Zielfunktionswert.

Bei mittlerer Nachfrage (Tabelle 6.11) ist die ADP-Strategie im Fall des kleinen Systems mit kleineren Stationen im Hinblick auf die Zielfunktion überlegen. Es wird mehr transportiert aber noch mehr an Kundenumwegen verhindert, sodass in Summe die Zielfunktion um knapp 10% besser abschneidet. Im kleinen System mit den größeren Stationen wird ebenfalls mehr transportiert, jedoch reichen die Einsparungen an Kundenumwegen nicht aus, um einen besseren Zielfunktionswert zu erreichen. Es werden folglich wieder zu früh oder auch unnötig Repositionierungen ausgelöst, was wiederum der Aggregation und den allgemein geringfügig auftretenden Umwegen zu Lasten zu legen ist. Auch im größeren System mit 100 Stationen ist die myopische Strategie mit der punktgenauen Reaktion bei leeren oder vollen Stationen überlegen. Bei 30 Stellplätzen pro Station erreicht sie in allen

Tabelle 6.10.: Vergleich der myopischen Puffer mit der ADP-Politik bei geringer Nachfrage

niedrige Nachfrage			**Zeit[$\frac{h}{Tag}$]**					**Anzahl**			
Anzahl		Politik	c^{tr}	c^{s}	c^{det}_{walk}	c^{det}_{bike}	gesamt	fehlende Räder	fehlende Boxen	Touren	repos. Räder
Stationen	Boxen										
36	30	myopisch (Puffer 0%)	**2,9**	**1,3**	**2,6**	**1,4**	**8,3**	**16,5 (1,5%)**	**13,2 (1,2%)**	**5,0**	**39,8**
		ADP	3,4	1,5	4,5	3,0	12,3	28,0 (2,6%)	28,2 (2,6%)	6,3	43,5
	60	myopisch (Puffer 0%)	0,9	0,6	**0,4**	**0,4**	**2,4**	**2,8 (0,3%)**	**3,6 (0,3%)**	1,5	18,4
		ADP	**0,4**	**0,3**	1,4	0,9	3,1	8,9 (0,8%)	8,7 (0,8%)	**0,8**	**9,7**
100	30	myopisch (Puffer 0%)	7,1	3,6	**6,4**	**2,3**	**19,4**	**67,1 (2,2%)**	**37,0 (1,2%)**	**8,9**	108,1
		ADP	**6,5**	**2,6**	13,9	8,1	31,1	146,1 (4,9%)	127,5 (4,3%)	11,1	**78,9**
	60	myopisch (Puffer 0%)	2,7	1,7	**1,1**	**0,5**	**6,0**	**11,3 (0,4%)**	**8,3 (0,3%)**	3,9	52,1
		ADP	**1,2**	**0,7**	3,0	2,1	7,0	32,0 (1,1%)	33,9 (1,1%)	**1,8**	**21,2**

Tabelle 6.11.: Vergleich der myopischen Puffer mit der ADP-Politik bei mittlerer Nachfrage

mittlere Nachfrage			**Zeit[$\frac{h}{Tag}$]**					**Anzahl**			
Anzahl		Politik	c^{tr}	c^{s}	c^{det}_{walk}	c^{det}_{bike}	gesamt	fehlende Räder	fehlende Boxen	Touren	repos. Räder
Stationen	Boxen										
36	30	myopisch (Puffer 0%)	**5,4**	**2,6**	15,0	5,0	28,0	94,2 (4,4%)	47,5 (2,2%)	**5,0**	**76,6**
		ADP	8,3	4,1	**9,9**	**2,9**	**25,2**	**62,0 (2,9%)**	**27,3 (1,3%)**	10,8	124,4
	60	myopisch (Puffer 0%)	**2,9**	**1,8**	3,4	1,9	**10,0**	21,4 (1,0%)	17,9 (0,8%)	**4,1**	**54,8**
		ADP	5,5	3,2	**2,0**	**0,9**	11,6	**12,3 (0,6%)**	**8,7 (0,4%)**	7,3	95,2
100	30	myopisch (Puffer 0%)	**12,1**	**6,6**	**24,6**	**8,6**	**51,9**	**259,3 (4,3%)**	**135,7 (2,3%)**	**13,4**	**198,0**
		ADP	27,0	12,4	25,5	9,9	74,7	267,6 (4,5%)	155,8 (2,6%)	29,0	370,6
	60	myopisch (Puffer 0%)	7,8	4,7	**5,7**	**3,2**	**21,4**	**59,6 (1,0%)**	**50,4 (0,8%)**	9,6	142,0
		ADP	**2,8**	**1,7**	15,7	10,6	30,7	164,7 (2,7%)	166,6 (2,8%)	**4,5**	**51,7**

Summanden der Zielfunktion niedrigere Werte. Bei 60 Stellplätzen wird zwar mehr Zeit für die Repositionierungsfahrten aufgewendet, dadurch aber mehr als 60% der Umwege eingespart und so eine deutlich geringere Gesamtzeit erreicht. Hier fällt wieder auf, dass bei ausreichend Puffer das ADP-Verfahren auf viele Repositionierungsaktionen verzichtet, da der bewertete Zustand mehr als einen Füllstand umfasst, aber bei der Bewertung nicht unterschieden wird.

Wird die Nachfrage weiter erhöht, verbessert sich die Performance des ADP-Ansatzes gegenüber dem myopischen. In drei der vier verschiedenen Instanzen wird zwar jeweils mehr Zeit in die Repositionierung investiert, jedoch werden so viele Umwege der Kunden verhindert, dass die Gesamtzeit, also der Wert unserer Zielfunktion geringer ausfällt. Nur im Fall des 100-Stationen-Systems mit 60 Stellplätzen, ist die Zeit für die Repositionierungstouren so hoch, dass die verminderten Kundenumwege nicht ausreichen, um einen besseren Zielfunktionswert zu erreichen. Dieser Effekt kann wieder auf die gleichen Probleme zurückgeführt werden, wie in den Fällen mit geringerer Nachfrage.
Es ist davon auszugehen, dass sich das antizipierende Verfahren weiter gegenüber dem myopischen verbessert, wenn die Nachfrageeigenschaften ausgeprägter sind, wie es im Fall der Praxisdaten erkennbar ist. Zum Beleg können weitere Experimente auf anderen Realdatensätzen oder auf realistischeren künstlichen Systemen dienen. Zu einer Performancesteigerung kann auch eine andere Einstellung des Aggregationsgrads bzw. des Optimierungsintervalls führen. Eine Ausweitung des Experimentumfangs in diesen Dimensionen war im Rahmen dieser Arbeit nicht zusätzlich möglich. Die jeweiligen Auswahlmöglichkeiten würden jeweils als Faktor die Anzahl der nötigen Simulationsdurchläufe erhöhen.

Tabelle 6.12.: Vergleich der myopischen Puffer mit der ADP-Politik bei hoher Nachfrage

hohe Nachfrage			**Zeit[$\frac{h}{Tag}$]**					**Anzahl**			
Anzahl		Politik	c^{tr}	c^{s}	c^{det}_{walk}	c^{det}_{bike}	gesamt	fehlende Räder	fehlende Boxen	Touren	repos. Räder
Stationen	Boxen										
36	30	myopisch (Puffer 0%)	**7,3**	**3,6**	51,7	10,9	73,6	325,1 (7,5%)	103,1 (2,4%)	**9,3**	**108,5**
		ADP	13,9	8,2	**35,2**	**6,7**	**63,9**	**221,1 (5,1%)**	**63,3 (1,5%)**	15,7	245,6
	60	myopisch (Puffer 0%)	**5,0**	**3,1**	16,0	5,2	29,2	100,4 (2,3%)	48,8 (1,1%)	**6,7**	**92,9**
		ADP	11,6	6,9	**7,3**	**2,4**	**28,2**	**46,1 (1,1%)**	**22,6 (0,5%)**	12,0	206,5
100	30	myopisch (Puffer 0%)	**17,6**	**10,3**	106,3	25,2	159,3	1118,6 (9,3%)	397,6 (3,3%)	**18,0**	**307,8**
		ADP	44,7	24,9	**72,7**	**16,2**	**158,5**	**765,3 (6,4%)**	**255,0 (2,1%)**	44,4	748,3
	60	myopisch (Puffer 0%)	**15,7**	**9,4**	31,3	12,0	**68,4**	329,7 (2,7%)	189,9 (1,6%)	**17,5**	**281,0**
		ADP	36,4	21,9	**20,5**	**9,8**	88,6	**215,7 (1,8%)**	**154,1 (1,3%)**	23,6	658,2

7. Fazit

BSS werden auf der ganzen Welt als „grüne Alternative“ im ÖPNV gehandelt. Die Anzahl von Städten mit BSS ist in den letzten Jahren stark angewachsen. Die Vorteile der Nutzung von Rädern gegenüber motorisierten Verkehrsmitteln treten nur zu Tage, wenn die Nutzerzahlen entsprechend groß sind. Neben einer guten Infrastruktur und günstigen Preisen ist ein reibungsloser Serviceablauf ausschlaggebend für eine hohe und wachsende Nutzerzahl. Reibungsloser Service bedeutet im BSS-Kontext, dass ein Kunde an seiner Startstation ein Rad erhält und dieses an seiner Wunschzielstation wieder abgeben kann. Dies kann nicht immer garantiert werden, da durch die Stochastik und Dynamik im System eine Ungleichverteilung der Räder über die Stationen entsteht. Bei vollen oder leeren Stationen ist dann eine Rückgabe bzw. eine Entleihung nicht möglich. Durch Repositionierungen auf Betreiberseite können diese Situationen vermieden werden. Die Optimierung der Repositionierungsoperationen in BSS sind in der Literatur viel diskutiert (siehe Kap. 2.1.4). Keine der existierenden Ansätze deckt jedoch die herausgearbeitete Forschungslücke ab.

7.1. Zusammenfassung

Ziel dieser Arbeit war die Entwicklung eines Optimierungsverfahrens für das stochastische und dynamische Repositionierungsproblem in BSS realer Größe. Das Augenmerk lag dabei auf der taktischen sowie operativen Ebene. Strategische Maßnahmen wurden außen vor gelassen. Die Abläufe in einem BSS sind gut in einem Markovprozess (MDP) abzubilden (siehe Kap. 2.3.1). Dies erfolgte nach Definition von Systemzustand, Entscheidung und Zustandsübergang. Zur Modellierung des Repositionierungsproblems wurde eine Abwandlung des stochastischen und dynamischen IRP (siehe Kap. 3.1.3) herangezogen. Eine exakte Lösung ist für reale Systemgrößen nicht in polynomialer Zeit berechenbar. Daher wurde ein Dekompositionsansatz implementiert (siehe Kap. 5), der den taktischen und operativen Teil getrennt betrachtet. Die Ergebnisse der beiden Ebenen haben jedoch jeweils

Einfluss aufeinander. In dem hierarchisch dekomponierten Ansatz spiegeln Optimierung, Steuerung und Betriebsablauf die Ebenen eines Unternehmens wider. Die Informationen aus dem Betriebsablauf gelangen über einen Aggregationsschritt in der Steuerung zur Optimierung. Dort werden sie verarbeitet und nehmen über einen Disaggregationsschritt wieder Einfluss auf den Betrieb. Um den Ansatz zu evaluieren, wurde die in Kapitel 5.1 vorgestellte Simulation eines BSS entwickelt. Die modular aufgebaute, eventbasierte Simulation ist in der Lage, Systeme jeder Größe mit verschiedenen Nachfrageverhalten nachzuempfinden. Verschiedene Repositionierungspolitiken können angewendet werden. Größen von Interesse werden gespeichert und ausgegeben. So kann die Performance der verschiedenen Politiken leicht vergleichen werden. In Kapitel 6 wurden Experimente auf Praxisdaten sowie verschiedenen Instanzen von künstlichen Daten durchgeführt.
Zusammengefasst liegt der Hauptbeitrag dieser Arbeit zur Forschung in folgender Liste:

- Beschreibung der Vorgänge eines BSS mittels MDP
- Modulare, eventbasierte Simulation eines BSS
- Modellierung des Repositionierungsproblems als 1-DSIRPPD
- Entwicklung eines antizipierenden Dekompositionsansatzes zur Lösung des dynamischen stochastischen Repositionierungsproblems auf realen Instanzgrößen
- Experimentelle Evaluation des Ansatzes

Der MDP eignet sich zur Beschreibung aller Vorgänge im BSS sehr gut. Stochastische Einflüsse von außen sowie geplante Aktionen von Betreiberseite sind in großem Umfang abbildbar. Bei Betrachtung der Simulation stechen zwei Eigenschaften heraus. Zum einen ist das die Eventbasiertheit, die kontinuierliche Simulation erlaubt und dabei im Vergleich zu zeitschrittbasierten die deutlich bessere Performance aufweist. Zum anderen ist das die Modularität, die flexibel Erweiterungen jeder Art erlaubt. Die Implementierung der Simulation umfasst die grundlegenden Aktivitäten in einem BSS. Dazu gehören Entleihungen, Rückgaben, Repositionierungen, deterministische Fahrtzeiten und Umwege. Nicht betrachtet wurden Diebstahl und Verschleiß sowie verschiedene stochastisch variierende Geschwindigkeiten. Das Hinzufügen von verschiedensten neuen Aktionen und Vorgängen ist jedoch aufgrund des modularen Aufbaus möglich. Die mathematische Modellierung als 1-DSIRPPD

bietet sich an, da es sich beim Repositionierungsproblem um eine Kombination von DSIRP und PDP handelt, wobei es nur ein Gut zu transportieren gibt, nämlich die Fahrräder. Da sich die Schwierigkeiten der einzelnen Probleme bei der Kombination überlagern, ist es notwendig, eine Reduktion der Komplexität vorzunehmen. Dies geschieht in unserem Fall durch den eigens für das Problem entwickelten Dekompositionsansatz. Durch die Aggregation und Disaggregation sinkt die Genauigkeit der Lösung, jedoch wird so die Rechenzeit gesenkt, sodass eine Lösung ermittelt werden kann. Da die implementierte BSS-Simulation die täglichen Vorgänge in einem BSS gut abbildet, eignet sie sich hervorragend zur Evaluation der Ergebnisse. Verschiedene Repositionierungspolitiken können gegenüber gestellt und verglichen werden.

7.2. Ausblick

Der vorgeschlagene Ansatz erhebt nicht den Anspruch ideal zu sein. Vielmehr handelt es sich um einen vielversprechenden und ausbaufähigen Prototypen. Es bleiben viele Stellschrauben, an denen Veränderungen vorgenommen werden können.

Simulation

Die BSS-Simulation in dieser Arbeit geht von einem idealen Kundenverhalten und verschleißlosen Fahrrädern aus. Dies entspricht insofern nicht der Realität, dass die Betreiber sich auch mit defekten Rädern oder Stationen oder gar mit Diebstählen auseinander setzen müssen. Solche Vorkommnisse mindern natürlich die Verfügbarkeit von Fahrrädern oder Stellplätzen. Um sie einzubinden, können die Räder und Stellplätze beispielsweise eine mit der Zeit wachsende Ausfallwahrscheinlichkeit bekommen. Zusätzlich kann ein Reparaturereignis in die Repositionierungstouren integriert werden, sodass beschädigte Objekte wieder instand gesetzt werden können. Eine weitere Maßnahme zur Erhöhung der Realitätsnähe ist das Einfügen eines stochastischen Faktors in die Fahrt- bzw. Laufzeiten. Damit kann abgedeckt werden, dass verschiedene Menschen sich auch mit verschiedenen Geschwindigkeiten fortbewegen.

Aggregation

Das Aggregationslevel spielt bei der Optimierung eine große Rolle. In der vorliegenden Arbeit wird das Augenmerk auf die Rechenzeit gelegt, sodass in relativ

kurzer Zeit viele Trajektorien durchlaufen werden können. Darum ist das Aggregationslevel grob gehalten. Es kann durch eine feinere Wahl der Aggregationsstufen an Genauigkeit gewonnen werden, wenn man eine längere Laufzeit in Kauf nimmt. Dies kann die Lösungsgüte erhöhen, da die Entscheidungen differenzierter getroffen sowie besser interpretiert werden können. Weitere Untersuchungen mit Variationen der Aggregationslevel können eine Übersicht darüber liefern, wie sich das Verhältnis der Rechenzeit zur Lösungsverbesserung verändert.

Routenumsetzung

Das Routing erfolgt mittels einer einfachen Heuristik, welche eine sehr geringe Laufzeit hat. Da das Routing mit jedem Durchlauf einer Trajektorie mehrfach durchgeführt wird, ist die Rechenzeit ausschlaggebend. Es ist trotzdem möglich, unter entsprechend steigendem Rechenaufwand ein komplexeres Verfahren zu implementieren. Erst kürzlich sind Short-Term Strategies für das IRP in BSS in der Literatur zu finden [9]. Auf deren Basis können die Routen kontinuierlich, sozusagen während der Fahrt, geplant werden. Dadurch können die Gesamtfahrtzeit und somit auch die verwendete Fahrtzeitabschätzung verbessert werden.

Optimierung

Das ADP-Verfahren in dieser Arbeit basiert auf einer Wertetabelle für jeden Zustand. Neue Ansätze basieren auf dynamisch aufgebauten Tabellen, mit denen die Speicherkapazitäten entlastet werden können [77].

Diese Punkte liefern jeder für sich genügend Material für neue Forschungsarbeiten und können jeweils oder in Kombination den Prototypen dieser Arbeit ausbauen. Zudem ist es sicherlich von Wert, weitere Forschungsarbeit in die Ausweitung der Experimente zu investieren. Sie kann sowohl neues Wissen über die verschiedenen Verhaltensweisen von verschiedenen BSS generieren, als auch tieferes Verständnis über die Funktionsweisen der verschiedenen Repositionierungspolitiken unter verschiedenen Umständen vermitteln.

A. Herleitungen

A.1. Zu Formel 5.1

$$
\begin{aligned}
p(t) &= \int_0^t \lambda_{it} e^{-\lambda_{it} x} dx \\
&= \lambda \left[-\frac{1}{\lambda} e^{-\lambda_{it} x} \right]_0^t \\
&= \left[-e^{-\lambda_{it} x} \right]_0^t \\
&= -e^{-\lambda_{it} t} + e^0 \\
&= 1 - e^{-\lambda_{it} t} \\
\Leftrightarrow \quad e^{-\lambda_{it} t} &= 1 - p(t) \\
\Leftrightarrow \quad -\lambda_{it} t &= \ln\left(1 - p(t)\right) \\
\Leftrightarrow \quad t &= \frac{\ln\left(1 - p(t)\right)}{-\lambda_{it}}
\end{aligned}
$$

B. Algorithmen

B.1. Verbesserungsverfahren zum Routing (zu Kapitel 5.3)

Algorithmus 4: `doAllInnerChanges(List<Tuple <Station, Integer»route, int numInnerSwaps)`

for $num = 1, ..., $ *numInnerSwaps* **do**
  for $i = 1, ...,$ *route.size()* **do**
    for $j = i + 1, ...,$ *route.size()* **do**
      Seien i^- und j^- die Vorgänger und i^+ und j^+ die Nachfolger von i und j in `route`.
      Sei $\texttt{route}_{ij}$ die neue Route, die sich durch Vertauschen von i und j ergibt.
      Speichere $diff_{ij} = c^{tr}_{i^-j} + c^{tr}_{ji^+} + c^{tr}_{j^-i} + c^{tr}_{ij^+} - c^{tr}_{i^-i} - c^{tr}_{ii^+} - c^{tr}_{j^-j} - c^{tr}_{jj^+}$,
      falls $\texttt{route}_{ij}$ wieder eine zulässige Route ist.
    end
  end
  Wähle (i^*, j^*) so, dass es $\min_{(i,j)} diff_{ij}$ löst.
end
return $\texttt{route}_{i^*j^*}$

Algorithmus 5: `optSwapBetweenRoutes(List<List<Tuple<Station, Integer>> routes, int numSwaps)`

for $num = 1, ..., numSwaps$ **do**
  for $r1 = 1, ..., routes.size()$ **do**
    for $i1 = 1, ..., routes.get(r1).size()$ **do**
      for $r1 = r1 + 1, ..., routes.size()$ **do**
        for $i2 = 1, ..., routes.get(r2).size()$ **do**
          Sei `amount`(r, i) die zu transportierende Menge des Auftrags an Position i in Route r.
          if $amount(r1, i1) = amount(r2, i2)$ **then**
            Setze den Auftrag $(r1, i1)$ an die Stelle von Auftrag $(r2, i2)$ und umgekehrt.
            Seien $r1_{neu}$ und $r2_{neu}$ die resultierenden Routen.
            Speichere $diff_{(r1,i1),(r2,i2)} =$ `RouteLength(r1)` + `RouteLength(r2)` − `RouteLength`$(r1_{neu})$ − `RouteLength`$(r2_{neu})$, falls $r1_{neu}$ und $r2_{neu}$ wieder zulässige Routen sind.
          end
        end
      end
    end
  end
  Wähle $((r1, i1)^*(r2, i2)^*)$ so, dass es $\max_{((r1,i1),(r2,i2))} diff_{(r1,i1)(r2,i2)}$ löst.
  Sei $\texttt{routes}_{(r1,i1)^*(r2,i2)^*}$ die durch den Tausch entstehende neue Routenliste.
end
return $\texttt{routes}_{(r1,i1)^*(r2,i2)^*}$

Algorithmus 6: `optSwapPairsBetweenRoutes (List<List<Tuple<Station, Integer>> routes, int numSwaps)`

for $num = 1, ..., numSwaps$ **do**
  for $r1 = 1, ..., routes.size()$ **do**
    for $i1 = 1, ..., routes.get(r1).size()-1$ **do**
      Sei $\texttt{amount}(r, i)$ die zu transportierende Menge des Auftrags an Position i in Route r.
      if $amount(r1, i1) = -amount(r1, i1 + 1)$ **then**
        for $r2 = 1, ..., routes.size()$ **do**
          if $r1 \neq r2$ **then**
            for $i2 = 1, ..., routes.get(r2).size()$ **do**
              Verschiebe die beiden Aufträge $(r1, i1)$ und $(r1, i1 + 1)$ an die Stelle $i2$ in $r2$.
              Seien $r1_{neu}$ und $r2_{neu}$ die resultierenden Routen.
              Speichere $diff_{(r1,i1),(r2,i2)} =$ $\texttt{RouteLength(r1)} + \texttt{RouteLength(r2)} -$ $\texttt{RouteLength}(r1_{neu}) - \texttt{RouteLength}(r2_{neu})$,
              falls $r1_{neu}$ und $r2_{neu}$ wieder zulässige Routen sind.
            end
          end
        end
      end
    end
  end
  Wähle $((r1, i1)^*(r2, i2)^*)$ so, dass es $\max_{((r1,i1),(r2,i2))} diff_{(r1,i1)(r2,i2)}$ löst.
  Sei $\texttt{routes}_{(r1,i1)^*(r2,i2)^*}$ die durch den Tausch entstehende neue Routenliste.
end
return $\texttt{routes}_{(r1,i1)^*(r2,i2)^*}$

Akronyme

1-DSIRPPD one-commodity dynamic and stochastic Inventory Routing Problem with pickups and deliveries

1-PDTSP one-commodity Pickup-and-Delivery Traveling Salesman Problem

1-PDVRP one-commodity Pickup-and-Delivery Vehicle Routing Problem

ACO Ant Colony Optimization

ADP Approximative Dynamische Programmierung

ALNS Adaptive Large Neighborhood Search

BSS Bike-Sharing-System

CP Constraint Programming

DDMS distributed decision making system

DSIRP dynamisches und stochastisches Inventory Routing Problem

IP Inventory Problem

IRP Inventory Routing Problem

IS Informationssystem

LNS Large Neighborhood Search

MILP Mixed Integer Linear Programming

MIP Mixed Integer Programming

ÖPNV Öffentlicher Personennahverkehr

PDP Pickpup and Delivery Problem

PDS Post-Decision-Zustand

PICAV Personal Intelligent City Accessible Vehicle

SA Simulated Annealing

SP Swapping Problem

TSP Traveling Salesman Problem

VNS Variable Neighborhood Search

VRP Vehicle Routing Problem

Literaturverzeichnis

[1] Vélib Paris, März 2015. `http://en.velib.paris.fr/` (13.03.2015).

[2] C. Archetti, L. Bertazzi, G. Laporte, and M. G. Speranza. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Science*, 41(3):382–391, 2007.

[3] R. E. Bellman. The theory of dynamic programming. Technical report, DTIC Document, 1954.

[4] R. E. Bellman. A markovian decision process. Technical report, DTIC Document, 1957.

[5] R. E. Bellman and S. E. Dreyfus. Functional approximations and dynamic programming. 1959.

[6] M. Benchimol, P. Benchimol, B. Chappert, A. De La Taille, F. Laroche, F. Meunier, and L. Robinet. Balancing the stations of a self service "bike hire" system. *RAIRO-Operations Research*, 45(01):37–61, 2011.

[7] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[8] L. Bertazzi, A. Bosco, F. Guerriero, and D. Lagana. A stochastic inventory routing problem with stock-out. *Transportation Research Part C: Emerging Technologies*, 27:89–107, 2013.

[9] J. Brinkmann, M. W. Ulmer, and D. C. Mattfeld. Short-term strategies for stochastic inventory routing in bike sharing systems. *Transportation Research Procedia*, 10:364–373, 2015.

[10] L. Caggiani and M. Ottomanelli. A modular soft computing based method for vehicles repositioning in bike-sharing systems. *Procedia-Social and Behavioral Sciences*, 54:675–684, 2012.

[11] L. Caggiani and M. Ottomanelli. A dynamic simulation based model for optimal fleet repositioning in bike-sharing systems. *Procedia-Social and Behavioral Sciences*, 87:203–210, 2013.

[12] E. M. Cepolina and A. Farina. A new shared vehicle system for urban areas. *Transportation Research Part C: Emerging Technologies*, 21(1):230–243, 2012.

[13] D. Chemla, F. Meunier, and R. Wolfler Calvo. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10(2):120–146, 2013.

[14] N. Christofides. The vehicle routing problem. *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle*, 10(1):55–70, 1976.

[15] L. C. Coelho, J.-F. Cordeau, and G. Laporte. Thirty years of inventory routing. *Transportation Science*, 48(1):1–19, 2014.

[16] L. C. Coelho and G. Laporte. The exact solution of several classes of inventory-routing problems. *Computers & Operations Research*, 40(2):558–565, 2013.

[17] L. C. Coelho, G. Laporte, and J.-F. Cordeau. *Dynamic and stochastic inventory-routing*. CIRRELT, 2012.

[18] E. Côme and L. Oukhellou. Model-based count series clustering for bike sharing system usage mining: A case study with the vélib'system of paris. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):39, 2014.

[19] C. Contardo, C. Morency, and L.-M. Rousseau. *Balancing a dynamic public bike-sharing system*, volume 4. CIRRELT, 2012.

[20] T. G. Crainic and G. Laporte. Planning models for freight transportation. *European Journal of Operational Research*, 97(3):409–438, 1997.

[21] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[22] M. Dell'Amico, E. Hadjicostantinou, M. Iori, and S. Novellani. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*, 45:7–19, 2014.

[23] P. DeMaio. Bike-sharing: History, impacts,models of provision, and future. *Journal of Public Transportation*, 12(4):41–56, 2009.

[24] L. Di Gaspero, A. Rendl, and T. Urli. A hybrid ACO+ CP for balancing bicycle sharing systems. In *Hybrid Metaheuristics*, pages 198–212. Springer, 2013.

[25] L. Di Gaspero, A. Rendl, and T. Urli. Constraint-based approaches for balancing bike sharing systems. In *Principles and Practice of Constraint Programming*, pages 758–773. Springer, 2013.

[26] L. Di Gaspero, A. Rendl, and T. Urli. Balancing bike sharing systems with constraint programming. *Working paper, submitted*, 2014.

[27] W. Domschke, A. Scholl, and S. Voß. *Produktionsplanung*. Springer Berlin Heidelberg, 2. edition, 1997.

[28] M. Dror and L. Levy. A vehicle routing improvement algorithm comparison of a "greedy" and a matching implementation for inventory routing. *Computers & Operations Research*, 13(1):33–45, 1986.

[29] Duden. Rechtschreibung, März 2015. `http://www.duden.de/node/808908/revisions/1394849/view` (03.03.2015).

[30] European Environment Agency. A closer look at urban transport – term 2013: transport indicators tracking progress towards environmental targets in europe. *EEA Report*, 11/2013, 2013.

[31] O. K. Ferstl and E. J. Sinz. *Grundlagen der Wirtschaftsinformatik*. Oldenbourg Wissenschaftsverlag, 6. edition, 2008.

[32] E. Fishman, S. Washington, and N. Haworth. Bike share: A synthesis of the literature. *Transport Reviews: A Transnational Transdisciplinary Journal*, 33(2):148–165, 2013.

[33] J. C. García-Palomares, J. Gutiérrez, and M. Latorre. Optimizing the location of stations in bike-sharing programs: a gis approach. *Applied Geography*, 35(1):235–246, 2012.

[34] M. Gendreau and J.-Y. Potvin. *Handbook of metaheuristics*, volume 2. Springer, 2010.

[35] Gewista Werbegesellschaft m.b.H. Citybike Wien, Juni 2014. http://www.citybikewien.at (12.06.2014).

[36] C. Gunes, W.-J. van Hoeve, and S. Tayur. Vehicle routing for food rescue programs: A comparison of different approaches. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 176–180. Springer, 2010.

[37] A. Hax and D. Candea. *Production and inventory management*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[38] H. Hernández-Pérez and J.-J. Salazar-González. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145(1):126–139, 2004.

[39] S. C. Ho and W. Szeto. Solving a static repositioning problem in bike-sharing systems using iterated tabu search. *Transportation Research Part E: Logistics and Transportation Review*, 69:180–198, 2014.

[40] M. I. Hosny and C. L. Mumford. Solving the one-commodity pickup and delivery problem using an adaptive hybrid vns/sa approach. In *Parallel Problem Solving from Nature, PPSN XI*, pages 189–198. Springer, 2010.

[41] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, pages 237–285, 1996.

[42] R. Klein and A. Scholl. *Planung und Entscheidung: Konzepte*. Vahlen, München, 2. edition, 2011.

[43] C. Kloimüllner, P. Papazek, B. Hu, and G. R. Raidl. Balancing bicycle sharing systems: An approach for the dynamic case. In *Evolutionary Computation in Combinatorial Optimisation*, pages 73–84. Springer, 2014.

[44] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[45] J.-R. Lin and T.-H. Yang. Strategic design of public bicycle sharing systems with service level constraints. *Transportation Research Part E: Logistics and Transportation Review*, 47(2):284–294, 2011.

[46] L. M. Martinez, L. Caetano, T. Eiró, and F. Cruz. An optimisation algorithm to establish the location of stations of a mixed fleet biking system: an application to the city of lisbon. *Procedia-Social and Behavioral Sciences*, 54:513–524, 2012.

[47] G. Martinovic, I. Aleksi, and A. Baumgartner. Single-commodity vehicle routing problem with pickup and delivery service. *Mathematical Problems in Engineering*, 2008, 2009.

[48] D. C. Mattfeld and R. Vahrenkamp. *Logistiknetzwerke: Modelle für Standortwahl und Tourenplanung*. Springer Gabler, 2. edition, 2014.

[49] S. Meisel. *Anticipatory Optimization for Dynamic Decision Making*. Springer.

[50] P. Mertens, F. Bodendorf, W. König, A. Picot, M. Schumann, and T. Hess. *Grundzüge der Wirtschaftsinformatik*. Springer-Verlag, 2012.

[51] P. Midgley. Bicycle-sharing schemes: Enhancing sustainable mobility in urban areas. *Commission on Sustainable Development*, 19, 2011.

[52] D. C. Montgomery, M. Bazaraa, and A. K. Keswani. Inventory models with a mixture of backorders and lost sales. *Naval Research Logistics Quarterly*, 20(2):255–263, 1973.

[53] C. Nobis. Car sharing as a key contribution to multimodal and sustainable mobility behavior – the situation of car sharing in germany. *Transportation Research Record: Journal of the Transportation Research Board*, (1986):89–97, 2006.

[54] NYC Bike Share. citibike New York, März 2015. `https://www.citibikenyc.com/` (13.03.2015).

[55] C. H. Papadimitriou. The euclidean travelling salesman problem is np-complete. *Theoretical Computer Science*, 4(3):237–244, 1977.

[56] Paul DeMaio and Russell Meddin. The Bike-sharing Blog, February 2015. `http://bike-sharing.blogspot.de` (04.02.2015).

[57] G. POLYA. *How To Solve It - A New Aspect of Mathematical Method*.

[58] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[59] W. B. Powell, M. T. Towns, and A. Marar. On the value of optimal myopic solutions for dynamic routing and scheduling problems in the presence of user noncompliance. *Transportation Science*, 34(1):67–85, 2000.

[60] J. Pucher and R. Buehler. *City cycling*. MIT Press, 2012.

[61] G. R. Raidl, B. Hu, M. Rainer-Harbach, and P. Papazek. Balancing bicycle sharing systems: Improving a vns by efficiently determining optimal loading operations. In *Hybrid Metaheuristics*, pages 130–143. Springer, 2013.

[62] G. R. Raidl, J. Puchinger, and C. Blum. Metaheuristic hybrids. In *Handbook of Metaheuristics*, pages 469–496. Springer, 2010.

[63] M. Rainer-Harbach, P. Papazek, B. Hu, and G. R. Raidl. Balancing bicycle sharing systems: a variable neighborhood search approach. In *Proceedings of the 13th European conference on Evolutionary Computation in Combinatorial Optimization*, pages 121–132. Springer, 2013.

[64] M. Rainer-Harbach, P. Papazek, G. R. Raidl, B. Hu, and C. Kloimüllner. Pilot, grasp, and vns approaches for the static balancing of bicycle sharing systems. *Journal of Global Optimization*, pages 1–33, 2013.

[65] T. Raviv and O. Kolka. Optimal inventory management of a bike-sharing station. *IIE Transactions*, 45(10):1077–1093, 2013.

[66] T. Raviv, M. Tzur, and I. A. Forma. Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, 2(3):187–229, 2013.

[67] V. Ricker and D. C. Mattfeld. Ein Dekompositionsansatz für die Repositionierung in Bike-Sharing-Systemen. In *Tagungsband der Multikonferenz Wirtschaftsinformatik 2014, Paderborn*, pages 1450–1457, 2014.

[68] V. Ricker, S. Meisel, and D. C. Mattfeld. Optimierung von stationsbasierten Bike-Sharing Systemen. In *Proceedings of MKWI 2012, Braunschweig*, pages 215–226. GITO mbH Verlag Berlin, 2012.

[69] F. Rothlauf. *Design of modern heuristics: principles and application*. Springer, 2011.

[70] C. Schneeweiss. *Distributed decision making*. Springer Science & Business Media, 2012.

[71] J. Schuijbroek, R. Hampshire, and W.-J. van Hoeve. Inventory rebalancing and vehicle routing in bike sharing systems. Tepper School of Business, Paper 1491, 2013. `http://repository.cmu.edu/tepper/1491` (20.05.2014).

[72] S. A. Shaheen, S. Guzman, and H. Zhang. Bikesharing in europe, the americas, and asia: Past, present, and future. *Transportation Research Record: Journal of the Transportation Research Board*, (2143):159–167, 2010.

[73] J. Shu, M. C. Chou, Q. Liu, C.-P. Teo, and I.-L. Wang. Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems. *Operations Research*, 61(6):1346–1359, 2013.

[74] J. Si, A. Barto, W. Powell, and D. Wunsch. *Handbook of learning and approximate dynamic programming*, volume 2. John Wiley & Sons, 2004.

[75] O. Solyali and H. Süral. A branch-and-cut algorithm using a strong formulation and an a priori tour-based heuristic for an inventory-routing problem. *Transportation Science*, 45(3):335–345, 2011.

[76] S. T. Tan. Beiträge zur dekomposition von linearen programmen. *Unternehmensforschung*, 10(3):168–189, 1966.

[77] M. W. Ulmer, D. C. Mattfeld, and F. Köster. Budgeting time for dynamic vehicle routing with stochastic customer requests. 2015. working paper.

[78] F.-J. Van Audenhove, O. Korniichuk, L. Dauby, and J. Pourbaix. The future of urban mobility 2.0 – imperatives to shape extended mobility ecosystems of tomorrow. 08 2014. http://www.adlittle.com/future-of-urban-mobility.html.

[79] P. Vogel. *Service Network Design of Bike Sharing Systems - Analysis and Optimization*. Springer Berlin Heidelberg, 2016.

[80] P. Vogel, T. Greiser, and D. C. Mattfeld. Understanding bike-sharing systems using data mining: exploring activity patterns. *Procedia-Social and Behavioral Sciences*, 20:514–523, 2011.

[81] S. Voß, A. Fink, and C. Duin. Looking ahead with the pilot method. *Annals of Operations Research*, 136(1):285–302, 2005.

[82] D. A. White and D. A. Sofge. *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptative Approaches*. Van Nostrand Reinhold Company, 1992.

[83] F. Zhao, S. Li, J. Sun, and D. Mei. Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Industrial Engineering*, 56(4):1642–1648, 2009.